The University of Jordan

Authorization Form

I, Aman Mowaffaq Hajja, authorize the University of Jordan to supply copies of my Thesis/ Dissertation to libraries or establishments or individuals on request, according to the University of Jordan regulations.

Signature:

Date: ٢٠١١/٥/١٠

الجامعة الأردنيـــة
كلية الدراسات العليا

التاريـــخ: / /

نموذج رقم (١٨)
اقرار والتزام بالمعايير الأخلاقية والأمانة العلمية
وقوانين الجامعة الأردنية وأنظمتها وتعليماتها
لطلبة الماجستير

أنا الطالب: أمان محمد حجة الرقم الجامعي: (٨٠٦٢١٤٠)

تخصص: نظم معلومات حاسوبية الكليـــــة: تكنولوجيا المعلومات

عنوان الرسالة: Semi-Supervised Techniques for Arabic Text Classification

اعلن بأنني قد التزمت بقوانين الجامعة الأردنية وأنظمتها وتعليماتها وقراراتها السارية المفعول المتعلقة باعداد رسائل الماجستير عندما قمت شخصيا" باعداد رسالتي وذلك بما ينسجم مع الأمانة العلمية وكافة المعايير الأخلاقية المتعارف عليها في كتابة الرسائل العلمية. كما أنني أعلن بأن رسالتي هذه غير منقولة أو مستلة من رسائل أو كتب أو أبحاث أو أي منشورات علمية تم نشرها أو تخزينها في أي وسيلة اعلامية، وتأسيسا" على ما تقدم فانني أتحمل المسؤولية بأنواعها كافة فيما لو تبين غير ذلك بما فيه حق مجلس العمداء في الجامعة الأردنية بالغاء قرار منحي الدرجة العلمية التي حصلت عليها وسحب شهادة التخرج مني بعد صدورها دون أن يكون لي أي حق في التظلم أو الاعتراض أو الطعن بأي صورة كانت في القرار الصادر عن مجلس العمداء بهذا الصدد.

توقيع الطالب:.................. التاريخ: ١٠/ ٥ / ٢٠١١

SEMI-SUPERVISED TECHNIQUES FOR ARABIC TEXT CLASSIFICATION

By

Aman Mowaffaq Abdullah Hajja

Supervisor

Dr. Khalil El-Hindi

Co-Supervisor

Dr. Thair Hamtini

This Thesis was Submitted in Partial Fulfillment of the Requirements for the Master's Degree of Computer Information System

Faculty of Graduate Studies

The University of Jordan

May, 2011

COMMITTEE DECISION

This Thesis/Dissertation (Semi-supervised Techniques for Arabic Text Classification) was Successfully Defended and Approved on 12/4/2011

<u>**Examination Committee**</u>	<u>**Signature**</u>
Dr. Khalil M. El-Hindi (Supervisor) Assoc. Prof. of Artificial Intelligence	----------------------------
Dr. Thair M. Hamtini. (Co-supervisor) Assist. Prof. of E-Learning	----------------------------
Dr. Nadim Ali Obeid (Member) Prof. of Artificial Intelligence	----------------------------
Dr. Moh'd Belal Al-Zoubi (Member) Prof. of Data Mining	----------------------------
Dr. Jehad Ahmed Al-Sadi (Member) Assoc. Prof. of Parallel Computing (Arab Open University)	----------------------------

DEDICATION

To the one who has been by my side from the very beginning,
the one who has been there for me every step of the way,
the one who had cheered me up when I was sad,
Who left me up when I was down,
Who encouraged me through this journey,
And paved my way with support and inspiration.
To my husband, Saleh.

To my beloved family,
My shining stars through the dark hours.
To my dear parents,
Your support and love were endless,
Your guidance and care led me to where I am today.
And my dear brothers,
I am strong because you are by my side.

And last but not least,
To my lovely daughter Layan
Who made the whole journey worthwhile.
Your smile brightens my day.

To you all, I dedicate this work.

ACKNOWLEDGEMENT

I would like to express my great gratitude and thanks to my advisor, mentor, and teacher, Dr. Khalil El-Hindi, who has been there to guide me through the entire journey. The support and guidance made all the difference, and I shall always be grateful. I would also like to thank my co-supervisor, Dr. Thair Hamtini, who spared no effort to help in every way possible.

I am grateful for all the efforts and support that were put in this work, and they shall never be forgotten.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
ART	Adaptive Resonance Theory
ECOC	Error-Correcting Output Codes
EM	Expectation Maximization
GUI	Graphical User Interface
k-NN	*k*-Nearest Neighbor
LSI	Latent Semantic Indexing
MAP	Maximum A Posteriori
ML	Maximum Likelihood
NLP	Natural Language Processing
RBF	Radial Basis Function
SOM	Self-Organization Map
SS	Semantic Smoothing
SVMs	Support Vector Machines
TF-IDF	Term Frequency-Inverse Document Frequency

SEMI-SUPERVISED TECHNIQUES FOR ARABIC TEXT CLASSIFICATION

By

Aman Mowaffaq Abdullah Hajja

Supervisor

Dr. Khalil El-Hindi

Co-Supervisor

Dr. Thair Hamtini

ABSTRACT

The huge amount of textual information and the increase in the number of online documents is one of the many reasons behind the need for automatic document classification. For our study, we chose to use the naïve Bayes algorithm (a machine learning algorithm) to perform the task of automatic text classification. This algorithm has proven to work well in the text categorization area and has been known to produce very good results, with high classification accuracy. We target Arabic text in our study because we have found no previous works that attempted to use the semi-supervised learning technique in Arabic text classification. This language, just like other languages, is rich and unique in its own way and has its own unique characteristics. The issues and characteristics of Arabic language are addressed in our study and the classifier was modified and adjusted to fit the needs of the language. We trained our classifier using both supervised and semi-supervised learning techniques. One of the main goals of this study was to see if the task of Arabic text classification will improve as a result of semi-supervised learning, and so after using each learning technique we tested our classifier for classification accuracy and documented the

results. Our data set consisted of 1893 documents written in Arabic, these documents fall into 9 categories: Computer science, Economics, Education, Engineer, Law, Medicine, Politics, Religion, and Sports. The data set was divided into three sets: labeled data, unlabeled data, and test data. 70% of the data set formed the labeled data, 20% of the data set formed the unlabeled data, and 10% formed the test data. Both labeled and test data sets were used in the supervised learning technique, while all three data sets were used in the semi-supervised learning technique. The results of our tests were measured in recall, precision, fallout and error rate. The recall value for the classification accuracy of the supervised learning technique was 77% and the one for the semi-supervised learning technique was 87%. The semi-supervised proved to have a better classification accuracy for Arabic text than the supervised one by a difference of 10% recall value. It was also proven that the accuracy of the semi-supervised learning technique could not be further improved if an extra number of documents were added to the unlabeled data set.

Chapter (1): Introduction

Machine learning is a scientific discipline that attempts to simulate the process of learning in humans and animals into machines. *Learning* as a concept is defined using phrases such as "to gain knowledge of, understanding of, or skill in, by study, instruction, experience," and "modification of behavioral tendency by experience." When it comes to machines, we could say that a machine learns whenever it changes its structure, program, or data (based on its inputs or in response to external information) in such a manner that its expected future performance improves (Nils, 1998).

For the learner, data can be seen as examples that illustrate relations between observed variables so the learner can take advantage of this data (also called *examples*) to capture characteristics of interest from the information presented by these examples. A major focus of machine learning research is to understand the process of automatically learning how to recognize complex patterns and make intelligent decisions based on observed data; however, the set of all possible behaviors given all possible inputs is too large to be covered by the set of observed data (called *training set*). For that reason, the learner should be able to generalize its knowledge based on the training set already seen to include new unseen data and produce useful outputs for them.

Machine learning usually refers to the changes in systems that perform tasks associated with *artificial intelligence* (*AI*), and therefore, could be considered to be a

branch of AI. Machine learning is extremely useful and important for many reasons, from which we mention the following (Nils, 1998):

- In many cases, we find that among large piles of data, important relationships and correlations are hidden. Machine learning methods can often be used to extract these relationships (*data mining*).
- The amount of knowledge available about certain tasks might be too large for explicit encoding by humans. Machines that learn this knowledge gradually might be able to capture more of it than humans would want to write down.
- Environments change over time, in addition to the fact that new knowledge about tasks is constantly being discovered by humans. Machines that can adapt to a changing environment would reduce the need for constant redesign.

Machine learning has successfully been used in many real-world applications. We list below some of the most common ones (Nils, 1998):

1- Speech recognition, which is the process of converting spoken words to text. Machine learning is used in all systems for speech recognition to train the system to recognize speech.

2- Computer vision, which is the science and technology of machines that are able to extract information from an image that is necessary to solve some task. Many current systems, from face recognition systems, to systems that automatically classify microscopic images of cells, are developed using machine learning because the resulting systems are more accurate than hand-written programs.

3- Natural language processing (NLP), in which text classification is one of the main problems. In text classification, machine learning is used to predict the correct classification (category) of a new unseen document based on the observations and knowledge extracted from previously seen documents (training set).

Among the many different areas in which machine learning is employed, text classification (also called categorization) is the one we are interested in in our study. Text classification is the task of sorting a set of documents into categories from a predefined set. Text classification is an old research area but it gained more interest in the 1990s because of the increasing amount of textual information; the number of online documents is becoming huge and getting larger each day. Manual manipulation of this huge amount of data is very expensive, time consuming, and needs human expertise that cannot be available 24 hours a day, thus the need for automatic classification. Automatic text classification helps reduce the time needed to classify hundreds, even thousands, of documents every day, and will also save on the expenses and efforts of human experts.

For automatic text classification, we use a machine learning algorithm. The objective is to train the classifier using a set of documents, called the training set, pre-classified (by human experts) so that the classifier would have enough knowledge and information to automatically classify a new unlabeled document. This basic approach in automatic text categorization is called the supervised learning approach; in this approach the classifier first trains using a set of labeled data and is then given a set of unlabeled documents to automatically classify based on the information it has gained

in the training process. A more advanced and recent approach is called the semi-supervised learning approach; in this approach, not only does the classifier train on labeled documents, but also uses unlabeled documents for training. The idea is to first manually classify a set of documents and let the algorithm train on them, the next step is to give the classifier a set of unlabeled data set and let the classifier classify them. After that, the unlabeled documents that the algorithm classified are added to the training set along with their labels and the classifier would re-train on this new larger training set. The importance of the semi-supervised approach comes from the fact that manual classification will be reduced to include only a small number of documents, saving time, expense, and effort – in addition to the fact that semi-supervised learning had proven to improve the accuracy of a supervised one (Nigam, et al., 2006).

Several machine learning algorithms have been used for text categorization: support vector machines, *k* nearest neighbor, naïve Bayes, and neural networks are some of the most common ones, most of which were found to work quite well in the area of text classification. In our study, we have chosen to use the naïve Bayes as our classifier. Naïve Bayes is a simple probabilistic classifier based on applying Bayes' theorem (from Bayesian statistics) with strong naïve independence assumptions; it assumes that the presence/absence of a particular word in/from a class (category) is unrelated to the presence/absence of any other word. Despite the fact that naïve Bayes is simple and makes over-simplified assumptions, it has proven to work well in many complex real-world situations, and has been known to produce very good results, with high classification accuracy (Duwairi, 2007) (Nigam et al., 2006). These reasons contributed to our decision to choose naïve Bayes to be our classifier.

Researches in the text categorization area were mainly restricted to English text (Nigam et al., 2006) (Joachims 1997) (Rennie and Rifkin, 2001) (Zhou et al., 2008). Many studies also included other European languages such as French, German, and Spanish (Ciravegna, et al., 2000) as well as Asian languages such as Chinese and Japanese (Peng et al., 2003) (He et al., 2003) (Tsay and Wang, 2004). As far as we know, not many researches covered Arabic text. We also found no previous works that attempted to use the semi-supervised learning technique in Arabic text classification. This field is obviously new and under constant development, hence the uniqueness of our study. That’s why it was very important to us to perform our study on Arabic text, in addition to the fact that Arabic is the language of Qur'an (the holy book of Islam) and our native spoken language. Arabic, just like any other language, is very rich and unique in its own way, with many special characteristics that distinguish it from other languages. Letters of the Arabic alphabet take different shapes depending on their position within the word (beginning, middle, or end) and on whether the letter is connectable to its next neighbor. In addition, Arabic uses diacritics which are little symbols that are placed above or below a letter and act as short vowels. These diacritics are very often suppressed in Arabic text, usually creating ambiguity and misunderstanding. The diversity in letter styles and the use of diacritics make parsing Arabic text a nontrivial task. Another difficulty is faced when one is dealing with Arabic text is the fact that, often times, a verb or noun in its root pattern is augmented with prefixes, infixes, and suffixes to refer to the time during which the action described by the used word occurred and whether the verb or noun is plural or singular, and whether the gender of the participants is masculine or feminine. This means that several words can be mapped into the same root, creating a

dimensionality problem for the feature vector (the vector in which all keywords of a document are stored) causing it to grow very large.

The main objectives and goals of our study are:

1- To test the naïve Bayes algorithm when used for text classification in a semi-supervised learning environment.
2- To test the accuracy of the naïve Bayes algorithm in classifying Arabic documents.
3- To determine to which extent the task of Arabic text classification will improve as a result of semi-supervised learning.
4- To see at what point the classifier will converge to its best performance.

As mentioned earlier, we chose to use the naïve Bayes algorithm. We program it using Java programming language. The performance of both supervised and semi-supervised approaches were compared for accuracy. Since Arabic text was targeted, all the documents we used in our study were written in Arabic. Our data set was collected from online forums, magazines, and newspapers. We had a total of 1893 documents that varied in length and writing style. These documents fall into 9 different categories: Computer Science, Economics, Education, Engineering, Law, Medicine, Politics, Religion, and Sports, with a different number of documents for each category. The data set was divided into three groups: the labeled training data, the unlabeled data, and the data set used for testing. The documents were

preprocessed by removing all stopwords, symbols, and digits, and light stemming was used by removing some of the prefixes and suffixes from the keywords.

For testing, we used the 10-fold cross validation. In this type of testing, we test our classifier 10 times, using a different set of test documents for each round. The aim of the 10-fold cross validation is to obtain more accurate results due to the fact that the testing is not exclusive to one fixed set of data. The accuracy of the classifier was expressed only in terms of recall for all experiments; however, it was expressed in terms of recall, precision, fallout, and error rate for the experiment with the final and best results.

The experiments we held in our study are divided into two groups; the first one included experiments that we performed under certain assumptions about the algorithm. In the second group of experiments, we had to alter and modify these assumptions because they did not work well for our study. The experiments of the second group gave very good and satisfactory results.

In each one of our experiments, we used both supervised and semi-supervised learning approaches; for the supervised learning approach we ran the classifier and trained it over the labeled data then tested it over the test data, whereas for the semi-supervised learning approach we ran the classifier and trained it over the labeled data, then let it classify and learn from the unlabeled data, and after that tested it over the test data. Multiple distributions of the corpus were tried, i.e. several values of the percentage of labeled data against the percentage of the unlabeled one were taken. This was done in order to find the best distribution that suits the number of documents

we have. Multiple different ideas were also presented and tried in order to improve the performance of the classifier. The results of all experiments (measured in recall) were documented and compared.

The experiment that produced the best results was held under the modified assumptions about the algorithm. The data set in this experiment consisted of 1893 document, divided into 70% labeled data, 20% unlabeled data, and 10% test data. The results of this experiment were measured in recall, precision, fallout and error rate. The recall value for the classification accuracy of the supervised learning method was 77% and the one for semi-supervised learning method was 87%. It was also proven that the accuracy of the semi-supervised learning method could not be further improved if it was given an extra number of documents to classify and learn from. And so this experiment (*Experiment 5*, section 4.2.2) was considered to be the main and most satisfactory outcome of our study.

The rest of the thesis is organized as follows:

Chapter 2: A description of the main issues addressed in this study, and reviews of the main works that are related to our study.

Chapter 3: A description of the characteristics of the Arabic language and the main issues of the language that are addressed in our study.

Chapter 4: A description of the methodology we used in our study, the experiments we held, and the results.

Chapter 5: The main conclusion of our study, and some promising suggestions for future work.

Chapter (2): Literature Review

This chapter clarifies the main issues addressed in this study. The first section includes: the definition of text classification, text classification approaches and techniques, and Arabic text classification. The second section includes a review of the main works that are related to our study are discussed and their main contribution to the field.

2.1 Overview

In this section, we elaborate on the main subjects of interest in our study. Sub-sections include the definition of text classification, text classification approaches, text classification techniques, and Arabic text classification.

2.1.1 Text Classification

Document classification (also called *categorization*) is a problem in information science. It is the task of assigning a category to a free-text document. The descision should be based on the contents of this document. Text classification can provide conceptual views of document collections and has important applications in the real world. Many examples can be mentioned (Scholarpedia, 2010):

- News stories, which are typically organized by subject categories (topics) or geographical codes.

- Academic papers which are often classified by technical domains and sub-domains.
- Patient reports in health-care organizations, which are often indexed from multiple different aspects.
- Spam filtering which is an extremely widespread application of text categorization. This application is concerned with email messages and the need to classify them into two categories of "spam" and "non-spam".

Document classification tasks can be divided into three sorts: supervised document classification where some external mechanism (such as human feedback) is the means for providing information on the correct classification for documents, unsupervised document classification, where the classification must be done entirely without reference to external information, or human involvement, but by using a text classification method. And finally, the semi-supervised document classification, where parts of the documents are labeled by the external mechanism and the rest is classified using one of the text classification methods. (Cogsys, 2010)

The next two sections explain more about the three sorts of document classification, and also bout the different methods used in text classification.

2.1.2 Text Classification Techniques

Text categorization approaches are divided into three techniques: supervised, unsupervised, and semi-supervised. All three are explained clearly in the next sub-sections.

2.1.2.1 Supervised Text Classification

Supervised learning is the machine learning task of inferring a function from supervised training data, which consists of a set of training examples. Supervised training data in text classification is a set of documents manually classified and decided to be of a relevance to a certain category or multiple categories. In supervised text classification, each example is a pair consisting of an input object (the document) and a desired output value (the category to which it belongs). A supervised leaning algorithm analyzes the training data and produces a function called a *classifier* (if the output is discrete, which is the case in text classification) or a *regression function* (if the output is continuous). This function should be able to predict the correct output value for any valid input object. This means that the algorithm is required to generalize its knowledge from the training data it had seen, to unseen situations. (Stackoverflow, 2011) In text classification, predicting the correct output value for an input object means predicting the category to which the input document belongs to.

In order to use supervised learning approach for text categorization, we first need to gather a training set consisting of a collection of documents, along with the corresponding outputs (categories). We also need to determine the input feature representation, features in document classification being words, and so typically the representation would be a feature vector. The next step is to determine the machine learning algorithm (method) to be used (see section 2.1.3), and then run the algorithm on the training set of documents. Finally, an evaluation of the accuracy of the algorithm should be held, done by running the algorithm over a test set of documents

that is separated from the training set, and evaluating the accuracy of the output (Stackoverflow, 2011).

2.1.2.2 Unsupervised Text Classification

Unsupervised learning in machine learning is a class of problem in which one seeks to determine how the data are organized. When using this approach for text classification, the algorithm is given only unlabeled examples, unlike supervised learning in which the algorithm is given only labeled examples (Ghahramani, 2004).

The most common form of unsupervised learning is *clustering*. The clustering hypothesis states that documents having similar contents belong to the same category. A fixed collection of text is clustered into groups or clusters that have similar contents. The similarity between documents is usually measured with the associative coefficients from the vector space model (Cogsys, 2010).

Among neural networks models, the Self-organization map (SOM) and Adaptive resonance theory (ART) are commonly used unsupervised learning algorithms (Ghahramani, 2004).

2.1.2.3 Semi-supervised Text Classification

Semi-supervised learning is a class of machine leaning text classification techniques that uses both labeled and unlabeled data for training – typically, a small amount of labeled data with a large amount of unlabeled data. If we want to place the semi-supervised learning approach, it would fall between unsupervised learning (the

algorithm is given no labeled training data) and supervised learning (the algorithm is given all labeled training data) (Zhu, 2008).

In order to have labeled documents, the help of a skilled human agent to manually classify the training examples is needed. The cost of the labeling process to get a fully labeled training data set is very high, whereas acquisition of unlabeled data set is not costly. Semi-supervised learning can be very helpful and practical in such situations, saving cost and effort (Zhu, 2008).

One approach used in semi-supervised learning is co-training, where two or more learners are each trained on a set of examples, but with each learner using a different set of features for each example. Another approach is to treat the labels (category name) of the unlabeled data as 'missing data' and then use a technique that handle missing data, such as Gibbs sampling or the EM algorithm (Zhu, 2008).

2.1.3 Methods Used in Text Classification

There are multiple different methods that can be used in text classification; these are those methods (Quora, 2011):

- Naive Bayes Classifier
- Support Vector Machines
- *k*-Nearest Neighbor
- Artificial Neural Network
- Decision Trees

The following sub-sections contain a brief explanation of each of the methods mentioned above.

2.1.3.1 Naïve Bayes Classifier

Naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem (from Bayesian statistics) with strong (naive) independence assumptions and is particularly suited when the dimensionality of the inputs is high (Rish, 2001) (Statsoft, 2010).

A naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature (Rish, 2001). In document categorization, this means that even though a document is considered to belong to a certain category because of the existence of multiple words in this document, a naïve Bayes classifier considers all of these words to independently contribute to the probability that this document belong to that certain category.

Despite their naive design and apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations, and can often outperform more sophisticated classification methods (Rish, 2001) (Statsoft, 2010).

An advantage of the naive Bayes classifier is that it requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the

variances of the variables for each class need to be determined and not the entire covariance matrix (Rish, 2001).

2.1.3.2 Support Vector Machines

Support vector machines (SVMs) are a set of related supervised learning methods. They analyze data and recognize patterns, and they are used for classification purposes. It was originally invented by Vladimir Vanpik (Vanpik, 1995) (Cortes, 1995).

What the SVM does is that it takes a set of input data, and then for each given input it predicts to which class the input belongs. In order to do that, the SVM training algorithm, given a set of training examples each marked as belonging to one of two categories, builds a model that predicts into which category the new example falls. This model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible, and then the new examples are mapped into that same space and predicted to belong to a category based on which side of the gap they fall on (Dtreg, 2011) (Welling,).

To be more formal, a hyperplane or a set of hyperplanes is/are constructed by the support vector machine in a high or infinite dimensional space. The hyperplane with the largest distance to the nearest training data points of any class produces a good

separation, which in turn produces a low generalization error as a classifier (Dtreg, 2011) (Welling,).

2.1.3.3 *K*-Nearest Neighbor Algorithm

K-nearest neighbor is one of the simplest machine learning algorithms. It is considered to be a type of instance-based learning, or lazy learning algorithm where the function is approximated locally and all the computations are postponed for classification time (Saravananthirumuruganathan, 2011)

K-NN is a method for classifying objects based on closest training examples in the feature space, this is how it works: when a new object is encountered, it is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors (k is s positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of its nearest neighbor. By neighbors we mean the objects taken from the training set of the algorithm (the training set being the set of objects for which the correct classification is known). Although in k-NN no explicit training step is required, this set of objects is still called a training set (Saravananthirumuruganathan, 2011).

2.1.3.4 Artificial Neural Network

An artificial neural network (ANN) is a mathematical/computational model that is inspired by the structure and functional aspects of biological neural networks (neural

networks of a human brain). What a neural network consists of is a group of artificial neurons, and the process of information is done using a connectionist approach to computation. In most cases, an ANN changes its structure based on changes in external or internal information that flows through the network during the learning phase, in which case it is called an adaptive system (Gershenson, 2003)

Artificial neural networks are inspired by the central nervous system and the biological neural networks that consist of: neurons, axons, dendrites and synapses. The biological NNs are mimicked in the artificial NNs. Similarities between the two exist, starting with the fact that the algorithm is designed to alter the strength (weights) of the connections in the network to produce a desired signal flow. Another similarity is that the functions in the algorithm are performed collectively and in parallel by the units, just as the brain neurons work (Gershenson, 2003).

2.1.3.5 Decision Tree Learning

Decision tree learning is a common method used in data mining and machine learning. The structure of decision trees (also called classification trees) consists of leaves, representing classifications, and branches, representing conjunctions of features that lead to those classifications (Mitchell, 1997).

The goal is to create a decision tree that works as a predictive model which maps observations about an item to conclusions about the item's target value based on several input variables. Interior nodes correspond to input variables. Leafs represent values of the target variable given the values of the input variables represented by the path from the root to the leaf (Mitchell, 1997).

2.1.4 Arabic Text Classification

Arabic text classification is a normal text classification (see 1.1.1) but done over Arabic text. Text categorization techniques that address Arabic text are rare in the literature, however, there are a few researches on that subject (Al-Harbi, et al., 2008) (El Halees, 2007) (Al Shalabi, et al., 2006) (Duwairi, 2007) (Duwairi, 2006). Most of the text categorization algorithms were tested against English text (Aas and Eikvil, 1999). In addition to English language there are many studies in European languages such as French, German, Spanish (Ciravegna, et al., 2000) and in Asian languages such as Chinese and Japanese (Peng, et al., 2003) (He, et al., 2003) (Tsay and Wang, 2004).

Text classification over Arabic text differs from that over English text, or any other language. Arabic language has its own characteristics that need to be considered when building a classifier, and those very characteristics are what make this language very special. For detailed information about Arabic language characteristics see Chapter 3.

2.2 Related Works

In this section, reviews of some of the most related works to our study are presented. Sub-sections include related works on supervised text classification, using both SVM and naïve Bayes algorithms, related works on semi-supervised text classification, and related works on Arabic text classification.

2.2.1 Related Works on Supervised Text Classification

In this sub-section, some related works in the area of supervised text classification are explained. We first demonstrate the works in which the SVM algorithm was used, then the ones in which the naïve Bayes algorithm was used. Both of these algorithms have been successfully used in text classification and are considered to be among the most successful ones.

Support Vector Machines (SVMs)

The authors of (Joachims, 1998) attempted to measure the performance of SVM when used in text classification. They argued that by looking at four properties of text, we would have theoretical evidence that SVMs should perform well for text categorization.

An experiment was held using the Reuters corpus to compare the performance of SVMs using polynomial and RBF kernels with four conventional learning methods commonly used for text categorization: density estimation using a naïve Bayes classifier (Joachims, 1997), the Rocchio algorithm (Rocchio, 1971), a distance

weighted *k*-nearest neighbor classifier (Mitchell, 1997) (Yang, 1997), and the C4.5 decision tree/rule learner (Quinlan, 1993).

The results of the experiment showed that the *k*-NN classifier performed best among the conventional methods. Compared to the conventional methods all SVMs perform better independent of the choice of parameters. Even for complex hypothesis spaces, like polynomials of degree 5, no overfitting occurred despite using all features. The approximated and microaveraged results are shown in table 2.1.

Table 2.1: Microaveraged performance of the four methods (Bayes, Rocchio, C4.5, and SVM) over all Reuters categories.

	Bayes	**Rocchio**	**C4.5**	**k-NN**	**SVM**
Microavg.	72.0	79.9	79.4	82.3	86.2

The experimental results of the experiment mentioned above showed that SVMs consistently achieve good performance on text categorization tasks, outperforming existing methods substantially and significantly. They concluded that SVMs are a very promising and easy-to-use method for learning text classifiers from examples.

Another research in the same area was proposed by (Rennie and Rifkin, 2001), in which the authors attempted to use the Support Vector Machines (SVMs) in conjunction with Error-Correcting Output Codes (ECOC) to improve the state-of-the-art in text classification.

The Error-Correcting Output Codes (ECOC) framework is a simple but powerful framework to deal with the multi-class categorization problem based on the

embedding of the binary classifiers. For more information on ECOC, see (Escalera et al., 2010).

The approach in (Rennie and Rifkin, 2001) was to apply ECOC to text classification with the linear SVM as the binary learner. According to the authors, ECOC has been applied to text classification with naïve Bayes as the binary learner with great success, and so the SVM provides an opportunity for improvement since it is known to perform binary classification better than naïve Bayes. Experiments were performed on the Industry Sector and 20 Newsgroups data sets and achieved lower error than ECOC with naïve Bayes, in addition to the lowest known error on both data sets. The authors of (Rennie and Rifkin, 2001) attributed the success of the SVM to its low binary error and its ability to generalize well when the distribution of the training examples is uneven (as is the case of Industry Sector).

The impact of the work done was great. It showed that the use of the SVM with ECOC can greatly improve multiclass text classification. It has shown that SVM improves performance over ECOC with naïve Bayes by up to 48% and achieves the lowest known error on two data sets. Also, the error that was achieved on Industry Sector was 85% lower than regular naïve Bayes. The authors of (Rennie and Rifkin, 2001) stated that training the SVM has been significantly slower than naïve Bayes, however, once the SVM was trained, classifying a document was no less efficient than naïve Bayes. ECOC with the linear SVM was found to be practical and usable in large classifier systems where efficiency is paramount.

Table 2.2 shows the results of multiclass classification experiments on the 20 Newsgroups and Industry Sector data sets, held by (Rennie and Rifkin, 2001). The top row of each table indicates the number of documents/class used for training. The second row indicates the binary classifier. The far left column indicates the multiclass technique. Entries in the table are classification error. The table shows that the SVM outperforms naïve Bayes (NB) across both datasets, all matrix types and all training set sizes.

Table 2.2: The results of multiclass classification experiments on the 20 Newsgroups and Industry Sector data sets.

20 Newsgroups	800		250		100		30	
	SVM	NB	SVM	NB	SVM	NB	SVM	NB
OVA	0.131	0.146	0.167	0.199	0.214	0.277	0.311	0.445
Dense 15	0.142	0.176	0.193	0.222	0.251	0.282	0.366	0.431
BCH 15	0.145	0.169	0.196	0.225	0.262	0.311	0.415	0.520
Dense 31	0.135	0.168	0.180	0.214	0.233	0.276	0.348	0.428
BCH 31	0.131	0.153	0.173	0.198	0.224	0.259	0.333	0.438
Dense 63	0.129	0.154	0.171	0.198	0.222	0.256	0.326	0.407
BCH 63	0.125	0.145	0.164	0.188	0.213	0.245	0.312	0.390

Industry Sector	52		20		10		3	
	SVM	NB	SVM	NB	SVM	NB	SVM	NB
OVA	0.072	0.357	0.176	0.568	0.341	0.725	0.650	0.828
Dense 15	0.119	0.191	0.283	0.363	0.461	0.542	0.738	0.805
BCH 15	0.106	0.182	0.261	0.352	0.438	0.518	0.717	0.771
Dense 31	0.083	0.185	0.216	0.315	0.394	0.478	0.701	0.737
BCH 31	0.076	0.140	0.198	0.292	0.371	0.462	0.676	0.743
Dense 63	0.072	0.135	0.189	0.279	0.363	0.453	0.674	0.744
BCH 63	0.067	0.128	0.176	0.272	0.343	0.443	0.653	0.734

Naïve Bayes

The works of (Zhou, et al., 2008) address a common issue in Bayesian text classifiers which is referred to as data sparsity problem, especially when the size of

training data is very small. The authors argue that the frequently used Laplacian smoothing and corpus-based background smoothing are not effective in handling it. The method they used extracts explicit topic signature (e.g. works, multiword phrases, and ontology-based concepts) from a document and then statistically maps them into single-word features. Their work show that when the size of training documents is small, the Bayesian classifier with semantic smoothing not only outperforms the classifiers with background smoothing and Laplacian smoothing, but also beats the state-of-the-art active learning classifiers and SVM classifiers.

The authors of (Zhou, et al., 2008) explain that due to the data sparseness problem explained above, some terms appearing in testing documents may not appear in training documents of some classes, and so to prevent zero probability, one has to use some smoothing techniques which assign reasonable non-zero probability to those "unseen" terms. Laplacian smoothing, which simply adds one count to all terms in the vocabulary, is frequently used for Bayesian model smoothing, however, according to (Zhou, et al., 2008), a more effective smoothing method is what may be referred to as semantic smoothing. This method incorporates context and sense information into the language model.

An experiment was held using three dataset collections: 20-Newsgroups (20NG), Los Angeles Times (LATimes), and OHSUMED. 20NG is collected from twenty different Usenet newsgroups. LATimes contains news articles. OHSUMED consists of scientific abstracts collected from Medline, an on-line medical information database.

First, there was an evaluation of the semantic smoothing (SS) with 1% data for training. The performance is shown in table 2.3. SS significantly outperformed Laplacian smoothing, and background smoothing on all three collections. This verified the hypothesis that semantic smoothing is more effective than Laplacian smoothing and background smoothing for Bayesian text classifiers when the number of training documents is small and the data is sparse.

Table 2.3: Comparison of SS to Lap and Bkg where 1% of the documents are used for training.

Collection	Lap	Bkg	SS
OHSUMED	0.352	0.372	**0.413**
20NG	0.427	0.526	**0.613**
LATimes	0.525	0.538	**0.581**

According to the results shown in table 2.3, it can be said that the more sparse the data, the more effective the semantic smoothing. To further validate this finding, another experiment was conducted with as many as 33% data for training. With so many training documents, sparsity will not be a serious problem for most categories in the three testing collections. The results are shown in table 2.4.

Table 2.4: Comparison of SS to Lap and Bkg where 33% of the documents are used for training.

Collection	Lap	Bkg	SS
OHSUMED	0.660	**0.667**	0.665
20NG	0.771	0.802	**0.820**
LATimes	0.728	0.726	**0.729**

As expected, the semantic smoothing in the case of 33% training data was much less effective than in the case of 1% training data. Thus, emphasizing the importance of the new method proposed by the authors of (Zhou, et al., 2008) to solve the data sparsity problem.

2.2.2 Related Works on Semi-supervised Text Classification

Researches on using semi-supervised training in text classification are still fresh and the related works on this type of training are very few. The work of Tom M. Mitchell, among others, in (Nigam, et al., 2006) covered semi-supervised text classification through an iterative Expectation-Maximization (EM) technique. This technique uses both labeled and unlabeled data to train the classifier. The basic EM algorithm that was used is summarized below

* **Inputs**: Collections X_l of labeled documents and X_u of unlabeled documents.

* Build an initial naïve Bayes classifier, $\hat{\theta}$, from the labeled documents, X_l, only. Use maximum a posteriori parameter estimation to find $\hat{\theta} = \text{argmax}_\theta \, P(X_l|\theta) \, P(\theta)$

* Loop while classifier parameters improve, as measured by the change in $l(\theta|X,Y)$ (the log probability of the labeled and unlabeled data, and the prior):

  * **(E-step)** Use the current classifier, $\hat{\theta}$, to estimate component membership of each unlabeled document, i.e., the probability that each mixture component (and class) generated each document, $P(c_j|x_i; \hat{\theta})$

  * **(M-step)** Re-estimate the classifier, $\hat{\theta}$, given the estimated component membership of each document. Use maximum a posteriori parameter estimation to find $\hat{\theta} = \text{argmax}_\theta \, P(X, Y|\theta) \, P(\theta)$

* **Output**: A classifier, $\hat{\theta}$, that takes an unlabeled document and predicts a class label.

Figure 2.1: The basic EM algorithm for semi-supervised learning of a text classifier

In their experiments, they used the 20 Newsgroups text classification dataset, consisting of about 20,000 Usenet articles evenly distributed across twenty newsgroups. The documents were preprocessed by removing stopwords and by

scaling word counts of each document in order to have constant length for each document. For testing, 20% of the documents were chosen from each newsgroup. A sum of 10,000 documents formed the unlabeled dataset. The remaining documents represented the labeled dataset and were partitioned into non-overlapping groups. The naïve Bayes classifier was used in this experiment.

The task of the classifier was to classify an article into the newsgroup to which it was posted. The EM algorithm was used and the results were documented. Figure 2.2 shows the effect of using EM with unlabeled data on this data set. The vertical axis indicates average classifier accuracy on test sets, and the horizontal axis indicates the amount of labeled training data on a log scale. In order to compare the classification accuracy of traditional naïve Bayes (no labeled documents) with an EM learner that uses up to 10000 unlabeled documents they have used different number of labeled documents to train the classifier.

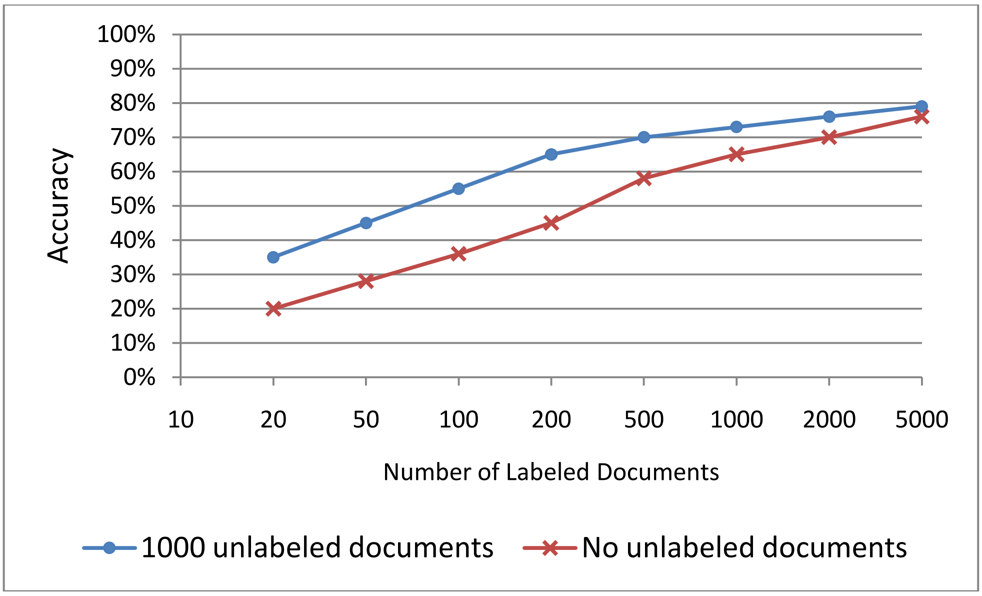


Figure 2.2: Classification accuracy on the 20 Newsgroups data set, both with and without 10,000 unlabeled documents.

As can be seen in the figure, the performance of EM showed a significantly better performance than that of traditional naïve Bayes. For example, with 300 labeled documents (15 documents per class), naïve Bayes reached 52% accuracy while EM achieved 66%. It can also be noticed that EM performs well even with a very small number of labeled documents; with only 20 documents (1 document per class), naïve Bayes obtains 20%, EM 35%. Another very important observation is that when the number of labeled data is high, EM does not outperform the traditional naïve Bayes; naturally, when there is already enough labeled data to accurately estimate the classifier parameters, having unlabeled data does not help nearly as much.

It was demonstrated and concluded that semi-supervised learning with labeled and unlabeled data provides text classifiers that are more accurate than those provided by supervised learning using only the labeled data.

2.2.3 Related Works on Arabic Text Classification

Text categorization techniques that address Arabic text classification are rare in the literature; however, it was addressed by a few researches. Following are some of them.

The authors of (Al-Harbi, et al., 2008) attempted to attain a better understanding of Arabic text classification, and to elaborate on it. The goal of the experiment that was held in this paper was to evaluate the performance of two popular classification algorithms (SVM and C5.0) in classifying Arabic text.

The data set in this work consisted of 17,658 documents taken from seven different Arabic corpora, split into 70% for training and 30% for testing. Results of the experiment, represented by the accuracy of both algorithms, are shown in table 2.5.

Table 2.5: SVM and C5.0 classification accuracy using seven Arabic corpora.

Classifier	Writer	SNP	Poems	Islamic	SPA	Web	Forums	Average
SVM	75.6%	72.7%	36.4%	86.4%	73.25%	68.67%	67.45%	68.65%
C5.0	86.4%	79.49%	49.15%	92.1%	79.8%	81.79%	80.1%	78.42%

As can be noted from the table, the C5.0 algorithm outperformed the SVM algorithm by about 10%. However, the authors of (Al-Harbi, et al., 2008) commented that a drawback of the C5.0 algorithm is that it is a black box algorithm that is only commercially available.

Another contribution is presented in the works of (El Halees, 2007), in which maximum entropy was used to classify Arabic documents. Their idea was to start by training the classifier by giving it the training set, after preprocessing the documents, and applying the maximum entropy algorithm, described in (Berger et al., 1996). The output of the trainer would be a training model which will be used to classify new documents. The categorizer will then classify new documents.

To experiment this method, the authors of (El-Halees, 2007) built an ArabCat system that classifies Arabic documents. The system was tested and its overall performance was compared with other existing systems. The results are shown in table 2.6.

Table 2.6: Performance of the ArabCat system compared to other systems.

System	Recall	Precision	F-Measure
ArabCat System	80.48	80.34	80.41
El-Halees	74.48	74.34	74.41
El-Kourdi et. al.	71.96	67.88	67.83
Sakhr's Categorizer	73.78	47.35	57.68
Sawaf et. al	84.2	50	62.7

From the table above, we can note that ArabCat system had better performance than that of other systems. And so this system holds potential in Arabic text categorization.

The *k*-nearest neighbor algorithm was used in (Al-Shalabi, et al., 2006) for Arabic text classification. The attempt was to evaluate the performance of this algorithm in classifying Arabic text. Their corpus consisted of 621 documents that belong to 6 categories. Documents in the data set were preprocessed by removing stop words and light stemming was done over the tokens. As for the *k* values of the algorithm, they used different values starting from 1 and ending with 30 in order to find the best results, and noticed that effectiveness started to decline at $k > 24$. The algorithm showed a result of 0.96 score in the experiment that they held, which was a very good result. However, they expected the scores to decline a little bit when using a larger number of categories.

The results of the work done by in (Al-Shalabi, et al., 2006) illustrated that *k*NN is applicable to Arabic text; very satisfactory results were achieved. It was also concluded that *k*NN can work well with a small number of training patterns provided that there is a sufficient number of examples for each category. In addition, the selection of the feature space, the training data set, and the value of *k* can enormously affect the accuracy of classification.

Rehab M. Duwairi contributed to the research in Arabic text classification in two of her papers. In (Duwairi, 2007), she compares the accuracy of three classifiers when used for Arabic text categorization. The first classifier is a distance-based one – where every category is represented as a vector of keywords that appear collectively in the training document examples that are known to belong to that category. Afterwards, unclassified documents are classified based on their closeness to category vectors. The second classifier is the classical naïve Bayes classifier, and the third one is the *k*-nearest neighbor (*k*nn) algorithm.

The corpus that was used consisted of 1000 documents. These documents fall into 10 categories, with a 100 document per category. For every category, 50 documents were randomly specified and used for training and the remaining 50 were used for testing.

Figure 2.3 depicts the values recall, precision, error rate, and fallout for the three classifiers mentioned above. It is clear that the naïve Bays classifier has the best accuracy. The *k*nn classifier comes in the second place (when $k = 50$), and the worst classifier for this dataset was the distance-base classifier.

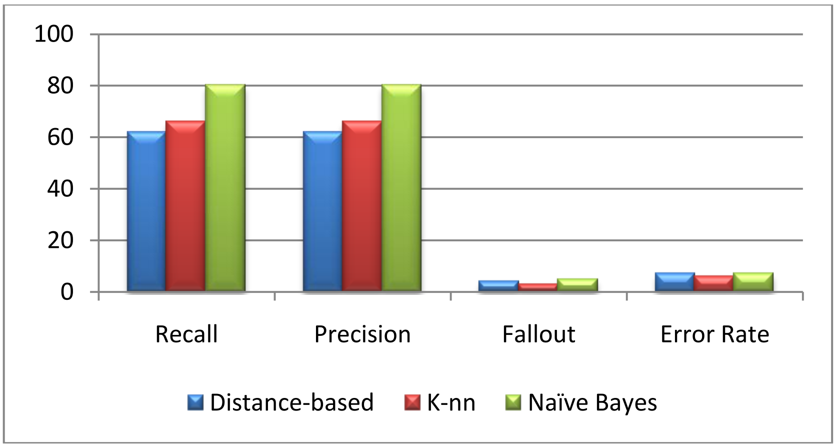


Figure 2.3: Recall, Precision, Fallout, and Error Rate for distance-based, knn, and naïve Bayes classifiers

In her other paper (Duwairi, 2006), Rehab M. Duwairi proposed a distance-based classifier technique in which categories are presented as feature vectors in an *m*-dimensional category space. Unclassified documents are classified on the basis of their closeness to category vectors. During training phase, the classifier scans the training documents *once* to extract category-specific features.

The model of the proposed distance-based classifier does not take word order into consideration. At the end of training, each category will be described as a feature vector in which words that appeared in documents and are known to belong to the category under consideration are stored.

The corpus is the same as that of (Duwairi, 2007). Table 2.7 shows recall, precision, fallout, and error rate for every category. It also shows the microaverage of these values for all categories. As shown in the table, the classifiers' accuracy was very good. It was concluded that the classifier is robust.

Table 2.7: Recall, precision, error rate, and fallout values for the 10 categories.

Category	Recall	Precision	Fallout	Error rate
Animal	0.6600	0.9167	0.0067	0.0400
Art	0.8400	0.3925	0.1444	0.1460
Economics	0.9800	0.3952	0.1667	0.1520
Internet	0.2200	0.6471	0.0133	0.0900
Medicine	0.6800	0.7727	0.0222	0.0520
Plant	0.5800	0.9355	0.0044	0.0460
Politics	0.4400	1.0000	0.0000	0.0560
Religion	0.5800	0.8529	0.0111	0.0520
Sport	0.9200	1.0000	0.0000	0.0080
Technology	0.3800	0.4872	0.0444	0.1020
Microaverage	0.6280	0.7400	0.0413	0.0744

We have found no previous works that attempted to use the semi-supervised learning technique in Arabic text classification. This field is obviously new and under constant development, hence the uniqueness of our study.

Chapter (3): Characteristics of Arabic

Arabic is a central semitic language, thus related to and classified alongside other semitic languages such as Hebrew and the Neo-Aramaic languages. Arabic has more speakers than any other semitic language. It is spoken by more than 280 million people, and is the official language of 22 countries. It is the language of Qur'an, the Holy book of Islam; see (Britannica, 2011).

3.1 Arabic Alphabet

The Arabic alphabet consists of 28 characters:

ا ب ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن هـ و ي

in addition to the Arabic *hamza* (ء), which is sometimes considered to be a letter. Three of the characters are vowels (ا و ي), while the rest of the letters are consonants. Arabic text is written from right to left. The characters of this language take different forms and shapes depending on two main things: first, their position within the word (beginning, middle, or end), and second, whether the letter is connectable to its next neighbor. Specifically, the following letters

ب ت ث ج ح خ س ش ص ض ط ظ ع غ ف ق ك ل م ن هـ ي

are connectable, i.e. connectable to their *next* neighbor, and the remaining ones

ا د ذ ر ز و

are not. For example, take a look at the different forms of the letter (هـ or *ha*) in the following cases:

- It has the shape (هـ) when it appears in the beginning of the word (such as the word (هرة) which means *cat*).
- It has the shape (هـ) when it appears in the middle of the word preceded by a non-connectable letter (such as the word (زهرة) which means *flower*).
- It has the shape (ـهـ) when it appears in the middle of the word preceded by a connectable letter (such as the word (نهر) which means *river*).
- It has the shape (ه) when it appears in the end of the word preceded by a non-connectable letter (such as the word (شِفاه) which means *lips*).
- It has the shape (ـه) when it appears in the end of the word preceded by a connectable letter (such as the word (منبَه) which means *alarm*).

The example we have chosen is an extreme case. Most of the other letters come in only two shapes: one shape is taken when the letter occurs at the beginning of the word or in the middle of the word (following a connectable or a non-connectable letter). The other shape is taken when it comes at the end of the word (following a connectable or a non-connectable letter). Some others come in three different shapes. It's also worth mentioning that there are several methods of writing Arabic scripts, and what we have just said refers to the most common one.

3.2 Diacritics

As mentioned earlier, Arabic has three vowels, which we should call *long vowels.* These correspond to the English vowels (a) (e) (o) and they are pronounced like (a) (ee) (oo) as in the words (back) (need) (soon). The Arabic symbols that indicate the pronunciations of (u) (oo) (e) in the words (sun) (look) (tell) are little symbols that are written above or below the relevant letters and are very often suppressed. Thus, one would write (book) and (buck) as (bk) where the context is to make clear which one is meant. However, if ambiguity is expected, the little symbols will then be used. Those little symbols are called *diacritics*, and the ones that indicate the pronunciations of (u) (ee) (o) are called *fat-ha*, *kasra*, and *damma* respectively. In addition, *shadda* is used to indicate the stressing of a letter. It can be said that *shadda* indicates the doubling of a letter, under the condition that both identical letters are pronounced with no diacritics.

In addition to the four diacritics mentioned above, *double fat-ha* (called *tanween fateh*), *double damma* (called *tanween damm*), *double kasra* (called *tanween kaser*), and *sukun* are used. Tha Arabic diacritics and their effects are shown below in Table 3.1.

Table 3.1: The Arabic diacritics and their effects.

Diacritic	**Effect**
◌ّ	Arabic *shadda*, which is used to double the letter on which it appears.
◌ُ	Arabic *damma*, which may appear above a letter and act as a short *wow* (و).
◌َ	Arabic *fat-ha*, which may appear above a letter and act as a short *alif* (ا).
◌ِ	Arabic *kasra*, which may appear below a letter and act as a short e (ي).
◌ْ	Arabic *sukun*, which may appear above a letter and means that the letter should be pronounced without any diacritics.
◌ٌ	Arabic *double damma*, which may appear above the last letter of a word and mean that the last letter of the word is pronounced as if it has *damma* followed by the sound of the letter (*n*).
◌ً	Arabic *double fat-ha*, which may appear above the last letter of a word and mean that the last letter of the word is pronounced as if it has *fatha* followed by the sound of the letter (*n*).
◌ٍ	Arabic *double kasra*, which may appear below the last letter of a word and mean that the last letter of the word is pronounced as if it has *kasra* followed by the sound of the letter (*n*).

In the computer, a letter with a diacritic is presented as two characters: one for the letter and one for the diacritic. Usually the position of a word in a sentence determines the diacritic of its last letter.

Some words in Arabic vary in meaning if the diacritics used in this word change. In this case, if diacritics are not added to clarify the meaning of the word, it is considered to be ambiguous. For more illustration on ambiguous words, we site the following examples:

- The word (شِفاه) means *lips*. Note that there is a *kasra* under the first letter (ش or *sheen*). When this *kasra* is replaced with a *fat-ha* placed above the letter, the word becomes (شَفاه) which means *cured*.
- The word (نَهْرْ) means *river*. Note that there is *sukun* above both letters (ـهـ or *ha*) and (ر or *ra*). When both *sukuns* are replaced with a couple of *fat-has*, the word becomes (نَهَرَ) which means *scolded*.
- The word (سَبَحَ) means *to swim*. Note that there is a *fat-ha* above the letter (ب or *ba*). When a *shadda* is added to the *fat-ha*, the word becomes (سَبَّحَ) which means *to praise*.
- The word (دَرَّبَ) means *to train*. Note that there is a *fat-ha* and a *shadda* above the letter (ر or *ra*). When both *shadda* and *fat-ha* are removed, the word becomes (دَرْب) which means *path*.

On the other hand, many of the words do not change in meaning due to change in diacritics. The change in diacritics in their case produces words that have no meaning, and so the meaning of the word is clear even if diacritics are suppressed. As an example, we mention the following:

- The word (سَرير) means *bed*. Note that there is a *fat-ha* on the first letter (س or *seen*). If this *fat-ha* is replaced with a *damma* or *kasra*, it becomes (سُرير) and (سِرير) respectively. Both words have no meaning.
- The word (مائِدة) means *table*. Note that there is a *kasra* below the letter (ئـ or *hamza*). If this *kasra* is replaced with a *fat-ha* or *damma*, it becomes (مائَدة) and (مائُدة) respectively. Both words have no meaning.

Unfortunately, the majority of Arabic text is written without diacritics. This is a big issue in the text classification problem, and leads to one of the complications of Arabic text categorization in contrast to the English language. The diacritics play a big role in the meaning of the word, however, most of the time they are ignored and omitted causing many words to lose their meaning or to be confused with other words. As a result, misclassifications are bound to happen, causing a decrease in the classification accuracy.

From what we have mentioned above in Sections 1 and 2, we can conclude that both different letter styles and diacritics make parsing Arabic text a nontrivial task.

3.3 Arabic Text Preprocessing

For the sake of categorization, we classify the contents of an Arabic text into keywords (nouns and verbs), pronouns, prepositions, conjunctions, auxiliary verbs, along with punctuation marks and formatting tags. For text categorization purposes, we need to extract the keywords from our Arabic documents because they are the ones that truly represent the content of the document, the rest of the words add no information that is useful in determining the type of the document. In fact, they could be harmful to the classification accuracy. The removal of all unimportant words from a document is called document filtering.

Arabic is a very rich language; often a verb in its root pattern is augmented with prefixes, infixes, and suffixes to refer to the time during which the event occurred and whether the verb is plural or singular (plural is divided into *two* and *three or more*;

both distinguishable in the Arabic language by the use of suffixes), as well as the gender of the participants in the verb. Let us take the verb (كتب or *kataba*) as an example, which means *wrote* in English. If the letter (ي or *ya*) is added as a prefix to the beginning of the verb, it becomes (يكتب or *yaktobo*) and the time of the verb becomes present, done by a singular male. But if, on the other hand, the suffix (ا or *aa*) is added to the verb, then it becomes (كتبا or *katabaa*) and means that the verb took place in the past, number of participants is two, and they are males. Table 3.2 below shows the different variations in the form of the verb (كتب or *kataba*) according to different factors.

Table 3.2: the different variations in the form of the verb (*kataba*).

Verb	Pronunciation in English	Time	Number of participants	Gender of subjects
كتبَ	Kataba	Past	One	Male
كتبتْ	Katabat	Past	One	Female
كتبا	Katabaa	Past	Two	Male
كتبتا	Katabataa	Past	Two	Female
كتبوا	Kataboo	Past	Three or more	Male
كتبنْ	Katabn	Past	Three or more	Female
يكتبْ	Yaktob	Present	One	Male
تكتبْ	Taktob	Present	One	Female
يكتبا	Yaktobaa	Present	Two	Male
تكتبا	Taktobaa	Present	Two	Female
يكتبونْ	Yaktoboon	Present	Three or more	Male
تكتبنْ	Taktobn	Present	Three or more	Female
سيكتبْ	Sayaktob	Future	One	Male
ستكتبْ	Sataktob	Future	One	Female
سيكتبانْ	Sayaktoban	Future	Two	Male
ستكتبانْ	Sataktoban	Future	Two	Female
سيكتبونْ	Sayaktoboon	Future	Three or more	Male

In addition to the variations mentioned above, a word in Arabic may also transform in a very unusual way by the use of diacritics, suffixes, prefixes and infixes. For example, the word (كَتَبَ or *kataba*) mentioned above is a verb in the past tense. If we change the diacritics of the first two letters from (*fat-ha* and *fat-ha*) to (*damma* and *kasra*), the word becomes (كُتِبَ or *kotiba*) and will become a verb in the past participant tense. If we add the prefix (م or *ma*) to it, it will become (مكتب or *maktab*), which is a noun that means *office*. If we add the prefix (م or *ma*) along with a suffix (ة or *ta marboota*) it will become (مكتبة or *maktaba*), which is a noun that means *library*. Also, if we add the infix (ا or *aa*) to the word, it will become (كاتِب or *kateb*), which is a noun that means *writer*. Fortunately, these variations do not affect the text classification accuracy in any way.

This richness in the Arabic language and the fact that there are numerous forms of the same verb contribute to its uniqueness and to the special characteristic that a single word (verb) can express the sense of the verb (past, present, or future), the number of participants in the verb, and the gender of the subject of the word. However, this same diversity in the forms of a verb is the cause of the dimensionality problem for the feature vector (the vector in which all keywords of a document are stored); a feature vector of a document can grow very large because of the different forms a word can take. Fortunately, all Arabic words can be mapped to their root patterns. Arabic words can have three-, four-, five-, or six-letter roots. More than 80% of Arabic words have three-letter roots (Duwairi, 2006). The process of extracting the root from a word is called *root stemming*. Stemming in general includes removing any added prefixes and suffixes to the word, and it is much needed in the text

classification problem for the purpose of reducing the dimensionality of the feature vector. According to (Al-Ameed, et al., 2005), there are two kinds of stemming:

1- Root stemming: a technique that attempts to reduce the word to its original root.
2- Light stemming: a technique that attempts to remove only some of the prefixes and/or suffixes. It does not attempt to remove any infixes or reduce the word to its root form (Al-Kharashi and Al-Sughayer, 2004).

Arabic language stemming takes a whole different form from that of the English language. According to (Duwairi, 2007), root stemming algorithms for Arabic text fall into two groups:

1- Algorithms that remove prefixes, infixes, and suffixes from words and then map them into a set of predefined root patterns. In this style, a word may need to be scanned several times before it can be mapped into its root pattern.
2- Algorithms that employ a letter weight and order scheme, where letters in a word are given weights and are assigned ranks or orders. These assigned weights and schemes are then used in extracting the root.

When using light stemming, some of the prefixes and/or suffixes should be removed. A list of some of the most common prefixes and suffixes in Arabic are listed in Table 3.3. Some or all of them can be removed using light stemming.

Table 3.3: Some of the most common prefixes and suffixes in Arabic.

Prefixes	Suffixes
و	ها
ال	ان
وال	ات
بال	ون
كال	ين
فال	يه
لل	ية
	ه
	ة
	ي

Choosing the kind of stemming and the technique to use is up to the researcher and depends on the objectives of stemming and the nature of the problem at hand. But in all cases, Arabic text stemming is essential and should not be skipped.

Chapter (4): Arabic Text Classification Techniques

This chapter presents the main work that we held in this study. The first section discusses the methodology that was used for Arabic text classification: an elaboration on the naïve Bayes classifier, the semi-supervised approach, our dataset and the issues related to it. The next section includes a detailed explanation of all the experiments we held in our study, a walk through all the steps we took in order to get the best results for the problem at hand, and the final results of all the experiments measured in the classifiers accuracy at classifying unseen documents.

4.1 Methodology

In this section, the methodology that we used in our study is presented. Sub-sections include the definition of the naïve Bayes classifier, how we implemented the algorithm and employed it in our work, an explanation of the semi-supervised learning approach, our data collection, the preprocessing that we performed on our documents, and the use of tokenizing and stemming.

4.1.1 Naïve Bayes Classifier

Naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem (from Bayesian statistics) with strong (naive) independence assumptions, it assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature (Resample, 2011). In document categorization, this means that even though a document is considered to belong to a

certain category because of the existence of multiple words in this document, a naïve Bayes classifier considers all of these words to independently contribute to the probability that this document belong to that certain category.

Naïve Bayes theorem is widely used as a text classifier because it has proven to give very good results in most of the researches that were done. Despite the fact that the independence assumptions it makes are incorrect, this classifier has proven to be a simple and efficient classifier.

Even though naïve Bayes is simple and makes over-simplified assumptions, it has proven to work very well in many complex real-world situations, and has also proven to outperform many other classifiers in the field of text classification. It has been known to produce very good results when used as a classifier, with high classification accuracy (Duwairi, 2007) (Nigam et al., 2006).

For all the reasons mentioned above, we have chosen to use the naïve Bayes classifier in our study.

The Algorithm:

Bayes' probabilistic theorem uses the following formula (Mitchell, 1997):

$$P(h/D) = \frac{P(D|h)P(h)}{P(D)} \quad (4.1)$$

Where:

$P(h)$ = prior probability of hypothesis h

P (D) = prior probability of training data D

$P(h/D)$ = probability of h given D

$P(D/h)$ = probability of D given h

This means that to find the probability of a certain hypothesis h given a set of data D, we need to calculate the probability of the data set D given the hypothesis h, multiply it by the prior probability of hypothesis h, and then divide the product by the prior probability of the training data D.

Usually, there would be multiple hypotheses, and we would want to know which would be the best to choose. The one with the highest probability given the data set would be the best one, naturally. And so, the purpose is to run the probability formula stated above over all hypotheses, and then choose the one with the highest probability, usually called *Maximum a posteriori* hypothesis h_{MAP}:

$$h_{MAP} = \arg\max_{h \in H} P(h|D)$$

$$= \arg\max_{h \in H} \frac{P(D|h)P(h)}{P(D)}$$

$$= \arg\max_{h \in H} P(D|h)|P(h) \quad (4.2)$$

If we assume that $P(h_i) = P(h_j)$, we can further simplify, and choose the *Maximum likelihood* (ML) hypothesis

$$h_{ML} = \arg\max_{h_i \in H} P(D|h_i) \quad (4.3)$$

Using formula 4.3, we can know which hypothesis has the highest probability amongst the collection of hypotheses, and therefore, which one is most suitable for the given data set at hand.

That was the basic form of Bayes theorem. The theorem was employed in the document categorization problem, with the training data being the collection of documents, known as the data set, and the hypotheses being the different classes (categories) to which these documents belong. The documents are modeled as sets of words where the (independent) probability that the *i*-th word of a given document occurs in a document from class C can be written as $P(w_i|C)$. We assume that the position of the word within the document is not relevant. Now the probability that a given document D contains all of the words w_i, given that it belongs to a class C, is:

$$P(D|C) = \prod_i P(w_i|C) \quad (4.4)$$

Now in order to employ the Bayes theorem in document classification, we need to know what the probability that a given document D belongs to a given class C is. In other words, we need to know the value of $P(C/D)$. By the definition of conditional probability in the case of statistical independence

$$P(D|C) = \frac{P(D \cap C)}{P(C)} \tag{4.5}$$

And

$$P(C|D) = \frac{P(D \cap C)}{P(D)} \tag{4.6}$$

Bayes' theorem manipulated these into this statement of probability:

$$P(C|D) = \frac{P(C)}{P(D)} P(D|C) \tag{4.7}$$

But because the probability of the document to be classified $P(D)$ will not change as we calculate the probability of this document being in each class, we can omit it from the equation, yielding the final form of the formula which was used in the Bayes classifier

$$P(C|D) = P(C)P(D|C) \tag{4.8}$$

The algorithm needs to go through two phases as a text classifier, the first phase is the learning phase, and the second one is the classifying phase. Both are explained next.

Phase 1: Learning

The first phase in naïve Bayes classifier is the *learning phase*, also known as the *Training phase*. In this phase, the algorithm is given a set of labeled documents, each document with its own label (the class to which it belongs). The roll of the learning phase is to use these labeled documents to infer useful information, and then use this information to make assumptions for the correct classification of unseen examples later on in the second phase.

Most of the work is really done in this phase; the preprocessing, the tokenizing, and the stemming (see 4.1.4, 4.1.5, and 4.1.6) are all done during this phase. The probability of a word being in each classification is calculated for all the words that appear in the training set (the bag of words called *vocabulary*). Those values will be used later in the second phase.

The pseudo code for the learning algorithm is (Mitchell, 1997):

Learn-Naïve-Bayes-Text (*Examples, V*)

1. Collect all tokens that occur in Examples

* *Vocabulary* ← all distinct words and tokens in Examples

* Filter *Vocabulary* by removing unimportant words, symbols, and digits

* Remove all words that appear <= 4 times from *Vocabulary*

* Do light stemming by removing some prefixes and suffixes from each word in *Vocabulary*

* Remove all diacritics from each word in *Vocabulary*

2- Compute all probabilities $P(v_j)$ *and* $P(w_k|v_j)$

* For each target value v_j in V do

- $docs_j \leftarrow$ *Examples* for which the target value is v_j

- $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$

- $Text_j \leftarrow$ concatenate all members of $docs_j$

- $n \leftarrow$ total number of words in $Text_j$ (counting duplicate words multiple times)

- for each word w_k in *Vocabulary*

  * $n_k \leftarrow$ number of times w_k occurs in $Text_j$

  * $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

Figure 4.1: Pseudo code for the naïve Bayes learning algorithm.

Where *Examples* represent the set of training documents, V represents the set of classes or categories $V = \{v_1, v_2 \ldots., v_i\}$, and $P(w_k|v_j)$ is the probability of a word being in each classification (Mitchell, 1997).

By the time this phase is complete, the classifier should have sufficient information to be able to predict the classification of a new unseen example. And so the second phase may start.

Phase 2: Classifying

The second phase in naïve Bayes classifier is the *Classifying* phase. In this phase, the algorithm is given a set of new unseen (unlabeled) documents, one at a time. The

algorithm then uses the Bayes formula 4.8 to predict the highest possible classification for the document, and assign this classification to it.

The pseudo code for the classifying algorithm is (Mitchell, 1997):

Classify-Naïve-Bayes-Text (*Doc*)

* *positions* ← all word positions in *Doc* that contain tokens found in *Vocabulary*

* Return v_{NB}, where

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_{i \in positions} P(a_i | v_j)$$

Figure 4.2: Pseudo code for the naïve Bayes classifying algorithm.

Where *Doc* is the new unlabeled document, and *positions* will contain all words seen in this new document that are also in *Vocabulary* (were seen in the training documents) (Mitchell, 1997).

The algorithm in this phase will calculate the probability value $P(Doc|v_i)$ for v_1, v_2 …., v_i. As mentioned before, V represents the set of classifications or categories. So the algorithm will calculate the probability of the document being in each different classification. After doing so, the algorithm will return v_{NB} which is the classification with the highest probabilities of all. This classification is then assigned to the document as the most probable classification.

The Implementation

As mentioned earlier, we used the naïve Bayes algorithm to be our classifier in this study. We programmed the algorithm using Java programming language. The GUI design of the program is shown in Figure 4.3.

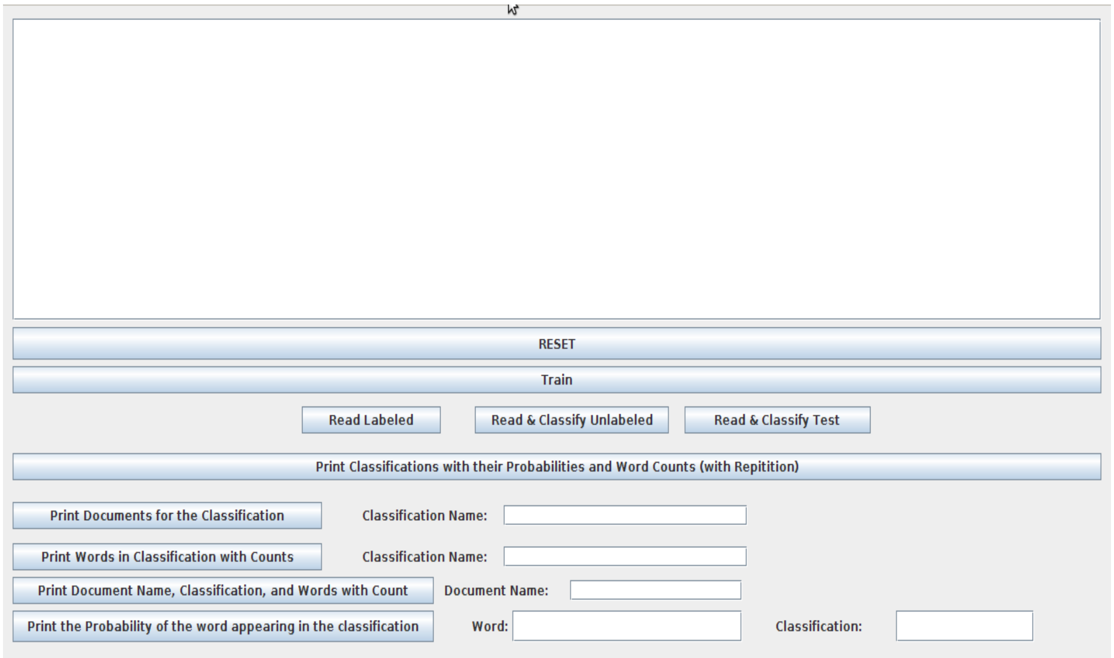


Figure 4.3: A screenshot of the GUI design of the program.

The "RESET" button is used to reset the program and clear the vectors and arrays in order to have a new start. The "Train", "Read Labeled", "Read & Classify Unlabeled", and "Read & Classify Test" buttons are used for the classification process. Any results are displayed in the big white box.

In order to use our classifier, we go through the following steps: The first step is to train the classifier using the labeled documents, and so "Read Labeled" button is used to read the labeled documents followed by the "Train" button which will start the training process. In case we are using the supervised learning approach, the next step

would be to click the "Read & Classify Test" button which will read the test documents and classify them and print out the results in the text area. However, in case we are using the semi-supervised learning approach, the next step would then be to click the "Read & Classify Unlabeled" button which will read the unlabeled documents and classify them, and then to click the "Train" button so that the classifier would retrain on the new set of documents (labeled + unlabeled), and finally click on the "Read & Classify Test" button. This way, we can compare between the performance of the classifier in a supervised learning environment against the one in a semi-supervised learning environment by comparing the output of clicking the "Read & Classify Test" the first time with the one the second time.

4.1.2 The Semi-supervised Approach

Semi-supervised learning is a class of machine leaning text classification techniques that uses both labeled and unlabeled data for training. If we want to place the semi-supervised learning approach, it would fall between unsupervised learning (the algorithm is given no labeled training data) and supervised learning (the algorithm is given all labeled training data) (Zhu, 2008)

The semi-supervised learning approach is a more recent approach than the supervised one. In this approach, the algorithm makes use of unlabeled data. It works like this: initially, just like a supervised approach, the algorithm is given a set of documents to learn from. After running the learning algorithm on these documents, the classifying algorithm is given a set of unlabeled documents to classify. The newly classified documents are then added to the pool of training documents and the

learning algorithm is run again over the new bigger set. So actually, the algorithm is learning from unlabeled data. It is also fair to say that the classifier is evolving as it sees new examples; the classifier would gain more knowledge every time it sees a new document and so the original classifier is bound to improve as new examples are encountered.

Tom M. Mitchell in (Videolectures, 2010) explained the process in a simple way. The two main steps in semi-supervised learning were called EM (Expectation Maximization)

EM: Repeat until convergence

1- Use training set (labeled data) to train the classifier

2- Apply the classifying algorithm to assign labeled to unlabeled data

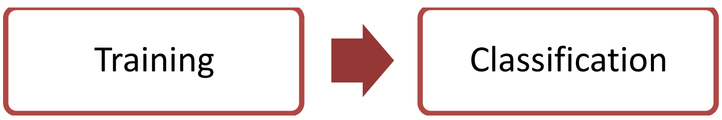
He elaborated more on the EM steps:

(E-step): Use the naïve Bayes classifier to estimate the classification for each unlabeled document.

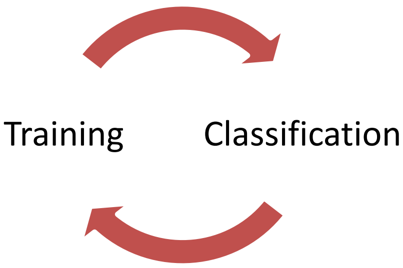
(M-step): Re-estimate the classifier given the new labeled documents.

For more details, see the basic EM algorithm in Section 2.2.2

A supervised approach goes through:



A semi-supervised approach goes through:



4.1.3 Data Collection

In order to run our classifier and test it, we needed to have a sufficient number of documents. Our data set, the Arabic text corpus, was collected from online forums, magazines and newspapers. We had a total of 1893 documents that vary in length and writing style. Those documents fall into 9 different categories: Computer Science, Economics, Education, Engineering, Law, Politics, Religion, and Sports, with a different number of documents for each category. The total number of documents for each classification is shown in Table 4.1. Each category was divided into three sets: labeled set (training set), unlabeled set, and the set used for testing.

Table 4.1: The number of documents in each of the 9 categories.

Category	# of Documents
Computer	120
Economics	270
Education	118
Engineer	165
Law	147
Medicine	283
Politics	232
Religion	277
Sport	282
Total	**1893**

Both labeled training data and test documents were used in the experiments of the supervised classifier, while the unlabeled data were used in the experiments of the semi-supervised classifier, along with the labeled and test data.

4.1.4 Document Preprocessing

Documents need to be processed before they are used by the classifier. Documents in their original form contain a collection of nouns, verbs, pronouns, prepositions, conjunctions, auxiliary verbs, along with punctuation marks and formatting tags. The meaningful words that truly express the content of a document are nouns and verbs and we will refer to them as *keywords*. The rest are unimportant words that need to be removed from all documents. This process is also called document filtering. Not filtering the documents will yield a huge vocabulary, one that will affect the efficiency of the classifier, and will harm the accuracy of the classification as well.

The words that were removed are all in Table 4.2

Table 4.2: The words removed from each document in the data filtering process.

ان	اما	الى	الذي	الخ	التي	الا	اذا	اذ	اخرى
انه	بل	بعض	بعد	بحيث	بانها	بانه	بان	اي	او
بما	خلال	حيث	حتى	ثمة	ثم	تلك	بين	بها	به
عليها	عليه	علينا	على	ذي	ذو	ذلك	ذاته	ذات	دون
كلها	كل	كذلك	قد	فيه	فيما	في	فمثلا	عند	عن
ما	لو	لهم	لهذه	لها	له	لما	لم	لا	كما
منا	من	ممكن	مما	معهم	معها	معه	معنا	مع	مثلا
هو	هناك	هنا	هل	هذه	هذا	منهما	منهم	منها	منه
					حين	يمكن	يجب	و	هي

In addition to words, our document filtering included removing some symbols, digits, and the English alphabets. The symbols are listed in Table 4.3

Table 4.3: The symbols removed from each document in the data filtering process.

)	(	*	&	^	%	$	#	@	!
>	،	<	.	/	©	+	=	-	_
:	"	~	{	}	’	،	؟	`	'
‘	÷	×	؛	|	\	[	]	-	/

For performance improvement purposes, we have also removed any word that appears ≤ 4 times in a document. Multiple experiments were held and we concluded that the removal of the words that appear ≤ 4 times improve the performance of the classifier (see 4.2.1).

4.1.5 Tokenizing

Tokenizing means breaking up a text document into tokens, or words. The purpose of tokenizing is to be able to manipulate individual words rather than the document as a whole. This is a very important step because our classifier considers words to be the features of the input documents, and word counts the feature values. The resulting output of tokenizing is a vector of individual words (keywords).

4.1.6 Stemming

Stemming is the process of reducing inflected (or sometimes derived) words to their stem, base, or root form (Comp, 2011). Stemming includes removing any added prefixes and suffixes to the word. Stemming can be one of two kinds (Al Amee et al., 2005):

1- Root stemming: a technique that attempts to reduce the word to its original root.

2- Light stemming: a technique that attempts to remove only some of the prefixes and/or suffixes. It does not attempt to remove any infixes or reduce the word to its root form (Al-Kharashi and Al-Sughayer, 2004).

Arabic language stemming takes a whole different form than that of the English language. According to (Duwairi, 2007), root stemming algorithms for Arabic text fall into two groups:

1- Algorithms that remove prefixes, infixes, and suffixes from words and then map them into a set of predefined root patterns. In this style, a word may need to be scanned several times before it can be mapped into its root pattern.

2- Algorithms that employ a letter weight and order scheme – where letters in a word are given weights and are assigned ranks or orders; and then the root is extracted by processing these assigned weights and ranks.

In addition to the difference in the technique of root stemming, a huge difference between Arabic and English is that Arabic uses diacritics (signals placed below or above letters to double the letter in pronunciation or to act as a short vowel (Duwairi, 2006)). (for more information on diacritics, See 3.2).

In our study, we have chosen to do some light stemming on the documents rather than root stemming. Our light stemming started by removing all the diacritics from our documents, those diacritics are listed below in Table 4.4

Table 4.4: The list of Arabic diacritics that we removed from our documents.

◌َ	Fat-ha *or* الفتحة
◌ُ	Damma *or* الضمة
◌ِ	Kasra *or* الكسرة
◌ّ	Shadda *or* الشدة
◌ْ	Sukun *or* السكون
◌ً	Double Fat-ha *or* تنوين الفتح
◌ٌ	Double Damma *or* تنوين الضم
◌ٍ	Double Kasra *or* تنوين الكسر

Next, we made the following replacements:

Table 4.5: The letters that were replaced in each document, and the letters they were replaced with.

The letter	Replaced with
أ	ا
إ	ا
آ	ا
ة	ه
ي	ى

And finally, we removed the following prefixes and suffixes from all the words:

Table 4.6: The list of prefixes and suffixes that were removed from the words in each document.

Prefixes	ال – وال – لل – بال – وبال – فال – كال – وكال
Suffixes	ها

4.2 Experiments and Results

In this section, all the experiments that we held in order to achieve the highest performance level on the algorithm are explained, along with their results. The first round of experiments was held under certain assumptions in the algorithm, whereas in the second round, these assumptions were changed and the algorithm was modified accordingly.

The experiments were performed using the naïve Bayes algorithm, which was programmed using Java programming language. We used both supervised training and semi-supervised training in every experiment. The goal was to compare between the results of the two approaches.

For testing, we used the 10-fold cross validation. We used the trained classifier to classify 10 sets of test documents, each of which include a totally different collection of documents and amounts to exactly 10% of the whole document count. The aim of this type of testing is to get more accurate results due to the fact that the testing is not exclusive to one fixed set of data. The 10 test results are then added and divided by 10 to get the average value. In order to get the 10 different sets of test data, we divided each category into 10 sets of documents, each represents 10% of the total documents in the category, and for each round of testing, we used one of these 10 sets as a test set and the rest were divided into two parts: labeled (training set) and unlabeled. This way, for each round of testing, we had a different set of documents to test our experiment.

The accuracy of the classifier is expressed only in terms of recall for all experiments, however, it is expressed in terms of recall, precision, fallout, and error rate for *Experiment 5* (section 4.2.2) , the experiment with the final and best results. For more elaboration on the formulae of the four terms consider a binary classification problem (i.e., there are only one category and *n* documents that need to be classified), so a given document either belongs to this category (i.e., positive example) or does not belong to that category (i.e., negative example). Assume that the classification is carried out by two classifiers: the first is a human and the second is a computer program. Then *recall (Re), precision (Pr), fallout,* and *error rate* are defined as

$$Re = \frac{a}{(a+c)}$$

$$Pr = \frac{a}{(a+b)}$$

$$Fallout = \frac{b}{(b+d)}$$

$$Error\ rate = \frac{(b+c)}{(a+b+c+d)}$$

Where *a* = number of documents that both the human and the computer classify as positive examples, *b* = number of documents that the human classifies as negative examples but the computer classifies as positive examples, *c* = number of documents

that the human classifies as positive examples but the computer classifies as negative examples, d = number of documents that both the human and the computer classify as negative documents, and $a + b + c + d = n$ (total number of test documents) (Duwairi, 2006).

According to the definition of semi-supervised training (see 4.1.2), this type of training takes an additional set of documents (unlabeled set), classifies them, and then retrains on them. Therefore, we expected that the semi-supervised training would perform better than the supervised one, and that it would improve the accuracy of the algorithm. It would seem quite simple, but the fact is: it's not! There are a number of factors and facts we need to consider here:

First of all, if the accuracy of the supervised method was not good, then we cannot expect the accuracy of the semi-supervised method to be better. This has a very simple explanation; the algorithm that performs poorly as a supervised method would naturally misclassify many of the unlabeled documents. These misclassified documents when added to the training set would be involved in the training process and would therefore cause the algorithm to decrease in accuracy. This would tell us that the classifier should have a good-enough performance to begin with, and that it should have good information and good knowledge as a supervised trainer in order for it to work better in a semi-supervised environment. That could be achieved by training the algorithm over a sufficient number of training documents.

Second, it is logical to think that the more documents that the algorithm trains on, the better the algorithm would be at classifying unseen data. This is true in a way, but

the fact is, the classifier does not keep getting better forever. At some point it will stop improving and will converge to its best performance. And this is what we tried to do in our study, which is to find at with point the classifier converges to its best performance, in addition to running all sorts of experiments in order to find the best factors and the best form of the algorithm that will produce the best results.

Another fact that was a key element in our study is that we are running our classifier over Arabic text. In order to achieve the good results we were hoping for, adjustments and modifications needed to be performed on the algorithm and that was not an easy task (see chapter 3).

4.2.1 Experiments Using the First Form of the Classifier

In this set of experiments the classifier would train on the training set (learn from the labeled data), then when it is given an unlabeled document, it will classify it, add it to the pool of documents with the same classification, then alter the values of the probabilities of the words seen in the new document that are also stored in *vocabulary* (the words that were seen previously in the "labeled" documents). Two main steps stand out in this set of experiments:

1- When a new word appears in an unlabeled document (a word that is not already stored in *vocabulary*), it will be ignored, and not added to vocabulary, and hence will not have a probability. This means that the vocabulary is constant, not dynamic. It

does not change when a new document is seen. Nevertheless, the probabilities of the words that were already in *vocabulary* will be modified.

2- The learning process from the unlabeled dataset is online, not batch (online learning means that the values are instantly updated as soon a new example is seen). It will learn from each document, one by one, and alter the values after each single document. This means that the classification of each document affects the classification of the document/s that will come after it.

At the beginning, we were under the assumption that adding the new words that appear in the unlabeled documents (words that were not already stored in vocabulary) to vocabulary in the retraining process would have a bad effect on the classifiers accuracy. Later on in Section 4.2.2 we discover and prove that this assumption is incorrect and that the step of adding new seen word to vocabulary in the retraining process is crucial for the improvement of the classifiers accuracy.

As explained in the second step mentioned above, we attempted to use online learning rather than batch learning. In section 4.2.2 we also prove that using batch learning is better than using online one for our study.

During the process of experimenting, we held yet another set of experiments for performance improvement purposes. These experiments were concerned with word counts in a document; we found that if a word appeared only a few times in a document, then adding this word to vocabulary and giving it a count and a probability will contribute nothing to the accuracy of the classifier, and would in fact decrease it. Removing those words from the document is the solution, but we needed to know just

the right word count that makes a word removable from a document. After performing multiple experiments, we found that removing a word that appears ≤ 4 times improves the accuracy. Keeping a word that appears less than 4 times will harm the performance of the classifier, as well as removing a word that appears more than 4 times.

Experiment 1

Initially, our corpus consisted of 1443 document. We thought we should try to divide them into the three sets as following:

70% Labeled data (training set) = 1010 document

20% Unlabeled data = 289 document

10% Test data = 144 document

The distribution of the documents over the 9 categories is shown in Table 4.7

Table 4.7: The number of labeled, unlabeled, and test documents taken from each category in experiment 1.

Category	Labeled	Unlabeled	Test	All
Computer	49	14	7	70
Economics	154	44	22	220
Education	47	14	7	68
Engineer	80	24	11	115
Law	68	19	10	97
Medicine	163	46	23	232
Politics	128	36	18	182
Religion	158	46	23	227
Sport	163	46	23	232
Total	**1010**	**289**	**144**	**1443**

We ran the classifier and trained it over the 70% labeled data, then tested it over the 10% test data (supervised training). Next we ran the classifier and trained it over the 70% labeled data, then let it classify the 20% unlabeled data, and after that tested it over the 10% test data (semi-supervised training). We used the 10-round testing and we documented the results (after taking the average of the 10 tests). They are shown in Figure 4.4 and Table 4.8.

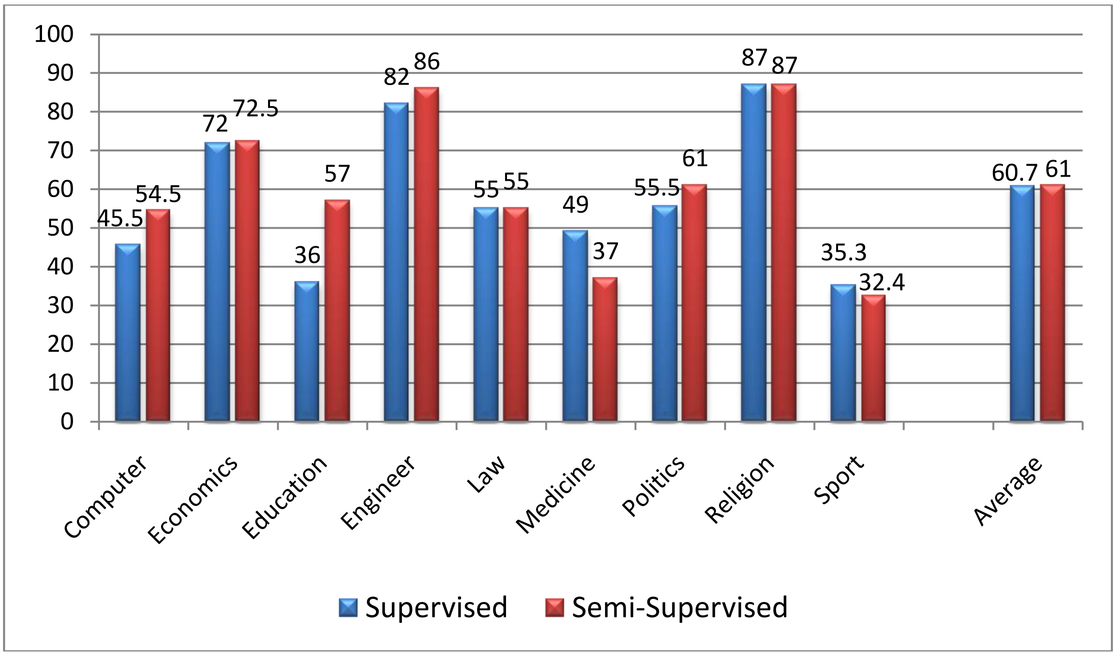


Figure 4.4: Results of experiment 1 for both supervised and semi-supervised methods.

Table 4.8: Results of experiment 1 for both supervised and semi-supervised methods.

	Supervised	**Semi-Supervised**
Computer	45.5%	54.5%
Economics	72%	72.5%
Education	36%	57%
Engineer	82%	86%
Law	55%	55%
Medicine	49%	37%
Politics	55.5%	61%
Religion	87%	87%
Sport	35.3%	32.4%
Average	60.7%	61%

As we can see from the results of this experiment, the accuracy of both supervised and semi-supervised training were law. In addition, the semi-supervised method did not improve the results compared to the supervised one. So we thought that maybe the labeled training document was not enough to build a good classifier, and therefore increased the percentage of the labeled documents.

Experiment 2

The corpus in this experiment was divided into the following three sets:

80% Labeled data = 1155 document

10% Unlabeled data = 144 document

10% Test data = 144 document

The distribution of the documents over the 9 categories is shown in table 4.9

Table 4.9: The number of labeled, unlabeled, and test documents taken from each category in experiment 2.

Category	Labeled	Unlabeled	Test	All
Computer	56	7	7	70
Economics	176	22	22	220
Education	54	7	7	68
Engineer	92	12	11	115
Law	77	10	10	97
Medicine	186	23	23	232
Politics	146	18	18	182
Religion	181	23	23	227
Sport	186	23	23	232
Total	**1155**	**144**	**144**	**1443**

We ran the classifier and trained it over the 80% labeled data, then tested it over the 10% test data (supervised training). Next we ran the classifier and trained it over the 80% labeled data, then let it classify the 10% unlabeled data, and after that tested

it over the 10% test data (semi-supervised training). We used the 10-round testing and we documented the results (after taking the average of the 10 tests). They are shown in Figure 4.5 and Table 4.10.

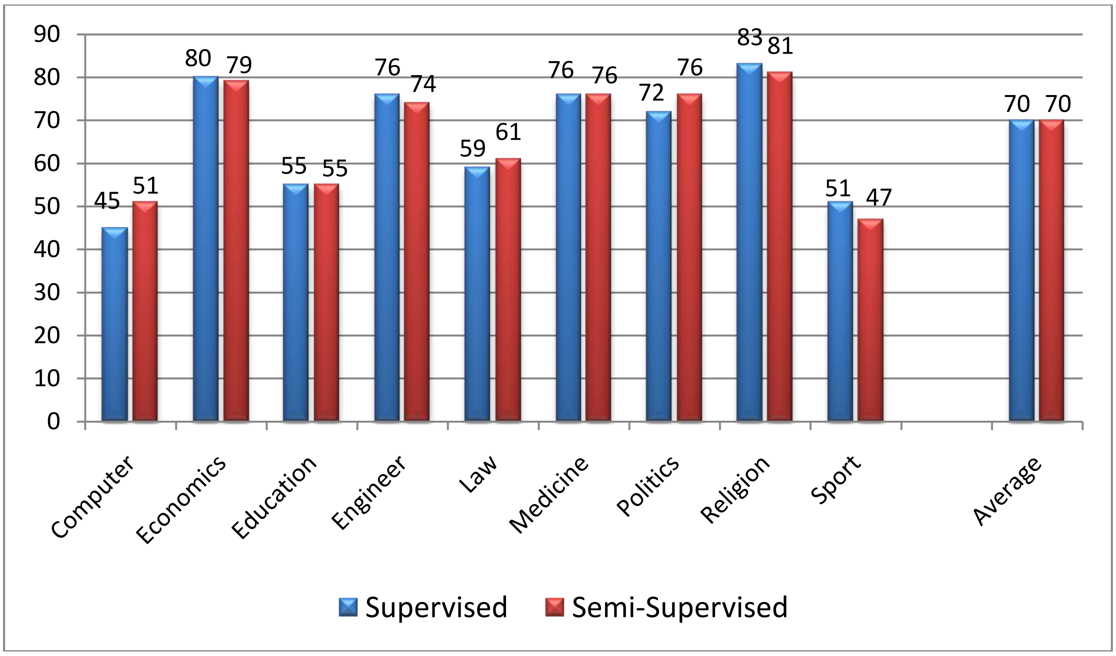


Figure 4.5: Results of experiment 2 for both supervised and semi-supervised methods.

Table 4.10: Results of experiment 2 for both supervised and semi-supervised methods.

	Supervised	**Semi-Supervised**
Computer	45%	51%
Economics	80%	79%
Education	55%	55%
Engineer	76%	74%
Law	59%	61%
Medicine	76%	76%
Politics	72%	76%
Religion	83%	81%
Sport	51%	47%
Average	70%	70%

As we can see, the results of both supervised and semi-supervised methods had improved, but we still did not see the effect of the additional 10% on the semi-supervised method. This was a problem because it was important to us that the unlabeled data that the algorithm classifies and learns from improve the classifier and increase the accuracy. So now we thought we should try and decrease the number of our categories.

Experiment 3

In this experiment, we thought we should try decreasing the number of categories and so we removed three of them. The three categories we chose to remove were the ones with the least number of documents, which were: Computer, Education and Law. In this experiment, the total number of the documents was 1208, and the distribution of the data was just like the previous one:

80% Labeled data = 966

10% Unlabeled data = 121

10% Test data = 121

Distributed over the 6 categories as follows:

Table 4.11: The number of labeled, unlabeled, and test documents taken from each of the 6 categories in experiment 3.

Category	Labeled	Unlabeled	Test	All
Economics	176	22	22	220
Engineer	92	11	11	115
Medicine	186	23	23	232
Politics	146	18	18	182
Religion	181	23	23	227
Sport	186	23	23	232
Total	**966**	**121**	**121**	**1208**

We ran the classifier and trained it over the 80% labeled data, then tested it over the 10% test data (supervised training). Next we ran the classifier and trained it over the 80% labeled data, then let it classify the 10% unlabeled data, and after that tested it over the 10% test data (semi-supervised training). We used the 10-round testing and we documented the results (after taking the average of the 10 tests). They are shown in Figure 4.6 and Table 4.12.

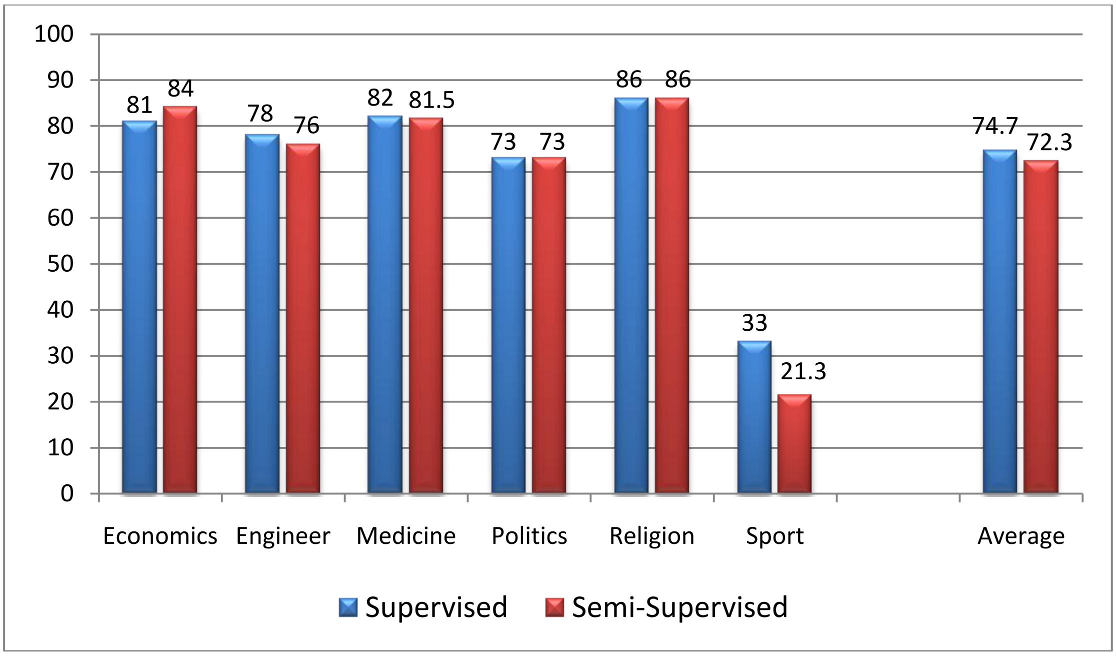


Figure 4.6: Results of experiment 3 for both supervised and semi-supervised methods.

Table 4.12: Results of experiment 3 for both supervised and semi-supervised methods.

	Supervised	**Semi-Supervised**
Economics	81%	84%
Engineer	78%	76%
Medicine	82%	81.5%
Politics	73%	73%
Religion	86%	85%
Sport	33%	21.3%
Average	74.7%	72.3%

The results of this experiment were also not satisfactory. The overall accuracy also may have increased a little bit, but we were still not able to achieve an improvement in the performance of the semi-supervised method that outperforms the supervised one. In fact, this time, the performance actually decreased a little. Here, we've started to think that maybe the number of documents we have is not enough to produce a good classifier in the first place, and that we should try to increase the initial number of documents we have (the corpus).

Experiment 4

For this experiment, we needed to increase the number of the documents in the dataset. In order to do so, we have downloaded an extra 450 documents from online articles, 50 new documents per category. The new corpus consisted of 1893 documents, which was divided into the 3 sets as following:

70% Labeled data = 1334

20% Unlabeled data = 370

10% Test data = 190

Distributed over the 9 categories as follows:

Table 4.13: The number of labeled, unlabeled, and test documents taken from each of the 9 categories in experiment 4.

Category	Labeled	Unlabeled	Test	All
Computer	76	32	12	120
Economics	196	47	27	270
Education	74	32	12	118
Engineer	112	37	16	165
Law	97	35	15	147
Medicine	207	48	28	283
Politics	166	43	23	232
Religion	201	48	28	277
Sport	206	48	28	282
Total	**1334**	**370**	**190**	**1893**

We ran the classifier and trained it over the 80% labeled data, then tested it over the 10% test data (supervised training). Next we ran the classifier and trained it over the 80% labeled data, then let it classify the 10% unlabeled data, and after that tested it over the 10% test data (semi-supervised training). We used the 10-round testing and we documented the results (after taking the average of the 10 tests). They are shown in Figure 4.7 and Table 4.14.

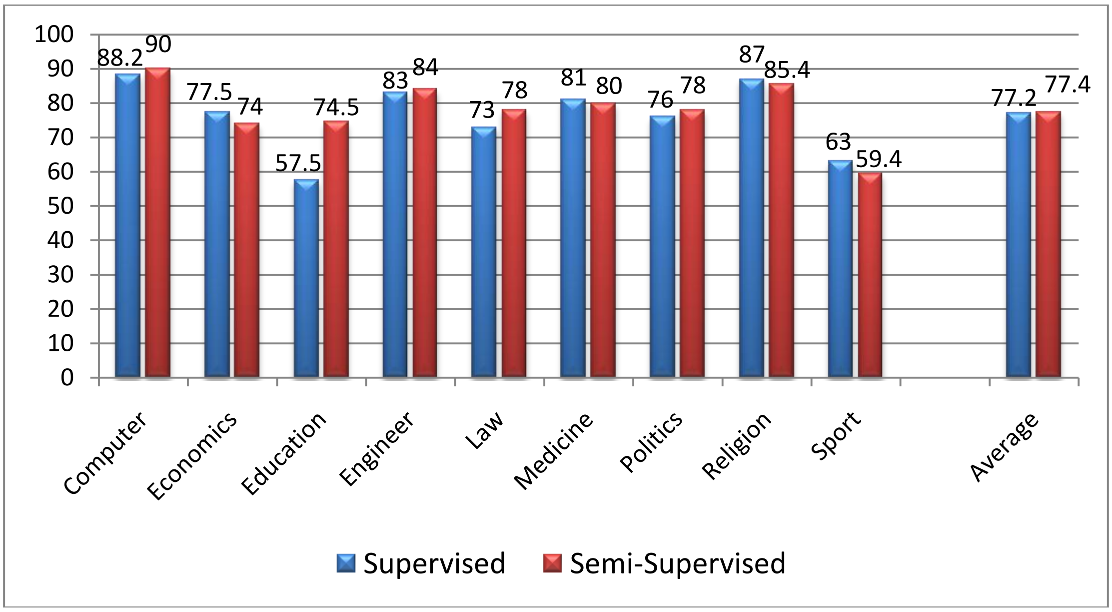


Figure 4.7: Results of experiment 4 for both supervised and semi-supervised methods.

Table 4.14: Results of experiment 4 for both supervised and semi-supervised methods.

	Supervised	**Semi-Supervised**
Computer	88.2%	90%
Economics	77.5%	74%
Education	57.5%	74.5%
Engineer	83%	84%
Law	73%	78%
Medicine	81%	80%
Politics	76%	78%
Religion	87%	85.4%
Sport	63%	59.4%
Average	77.2%	77.4%

The overall performance of the algorithm improved as a result of increasing the number of training data, a 77% accuracy seems like a good result. But again, the semi-supervised method is contributing nothing. The 20% that are classified by the algorithm and then added to the training set and used in retraining is doing nothing useful for the classifier and is not improving the accuracy. A 0.2% is really trivial. At this point, we thought that a change in the algorithm is needed, just a couple of changes in the way the algorithm works. It's all explained next.

4.2.2 Experiments Using the Second Form of the Classifier

In this set of experiments we attempted to deviate from the two assumptions mentioned in the previous section (4.2.1). In this set of experiments, the classifier learns from the labeled training data. Then when it is given an unlabeled document, it will classify it, and then add it to the pool of documents with the same classification. It will not alter any values yet. It does that to all unlabeled documents one by one and then when it is done seeing all unlabeled documents, it shall start the learning process all over again. It will not alter any values during the process of classifying unlabeled documents. We changed the two previous assumptions to become as follows:

1- When a new word is encountered in a new document, it is added to the vocabulary (the vocabulary becomes dynamic) and the new word will be given probabilities. In addition, the probabilities of the words that were already in vocabulary will be modified.

2- The learning process from the unlabeled dataset is batch, not online (batch means that the values remain constants until all examples are seen and then the values are modified all at one time); it will not learn as it sees each new document, instead, the learning process happens one time as the classifier finishes classifying all new documents. This means that the classification of each document will not affect the classification of the next seen document instantly.

The algorithm was reprogrammed to fit these new changes. And now new experiments were held in an attempt to achieve the improvement we are seeking.

Experiment 5

In this experiment, we were hoping that the changes we made to the algorithm would make a difference and improve the performance of the semi-supervised method. Our data set consisted of the 1893 document, and the distribution was just like the one in Table 4.13.

We ran the classifier and trained it over the 70% labeled data, then tested it over the 10% test data (supervised training). Next we ran the classifier and trained it over the 70% labeled data, then let it classify the 20% unlabeled data, and after that tested it over the 10% test data (semi-supervised training). We used the 10-round testing and we documented the results (after taking the average of the 10 tests). The results, expressed in terms of recall, precision, fallout and error rate are shown in Figures 4.8, 4.9, 4.10, and 4.11 and Table 4.15.

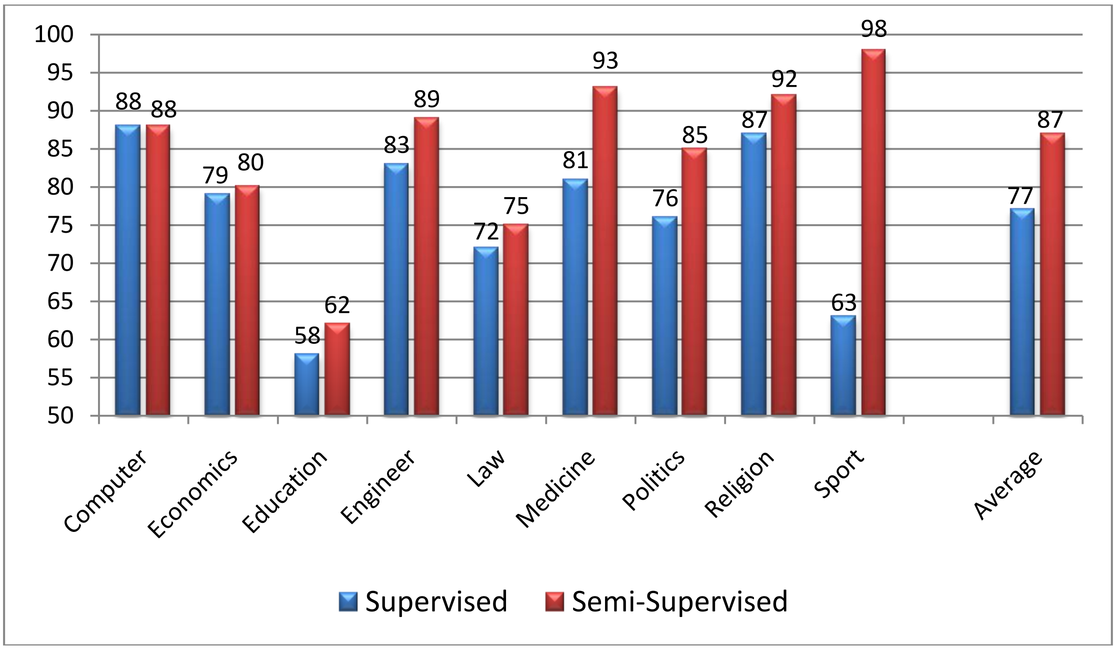


Figure 4.8: Results of experiment 5 for both supervised and semi-supervised methods (recall).

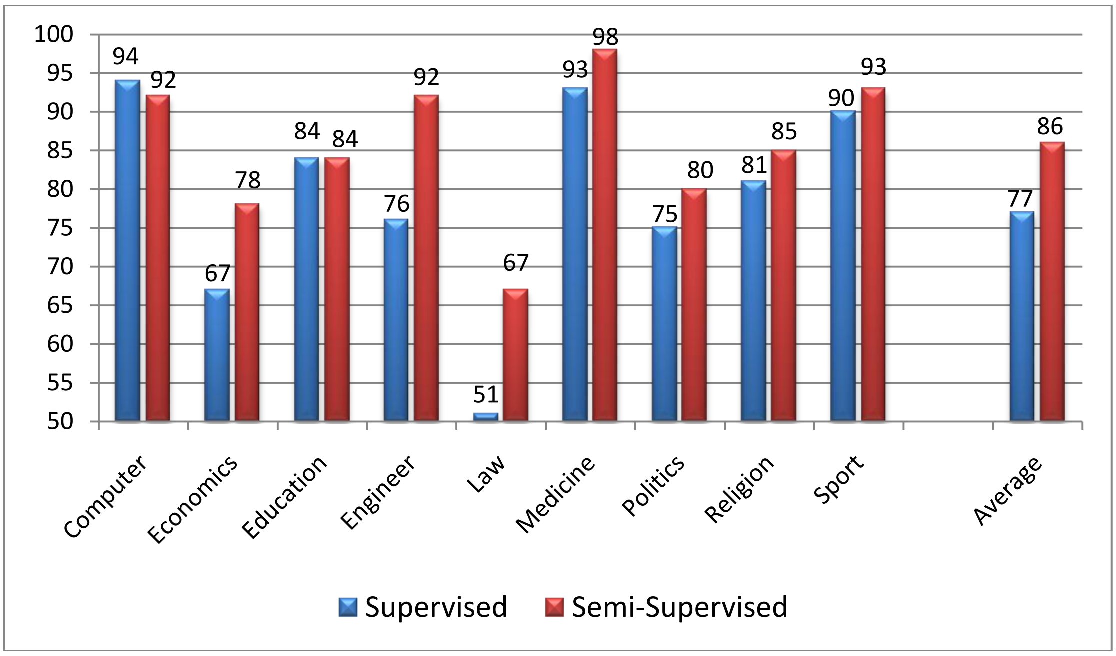


Figure 4.9: Results of experiment 5 for both supervised and semi-supervised methods (precision).

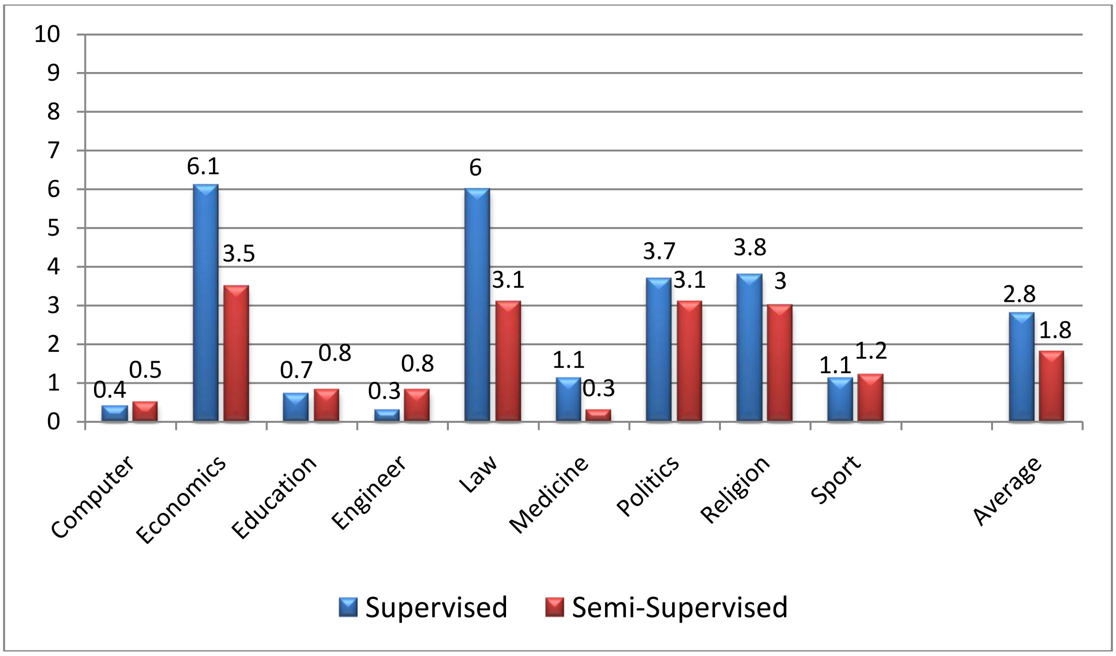


Figure 4.10: Results of experiment 5 for both supervised and semi-supervised methods (fallout).

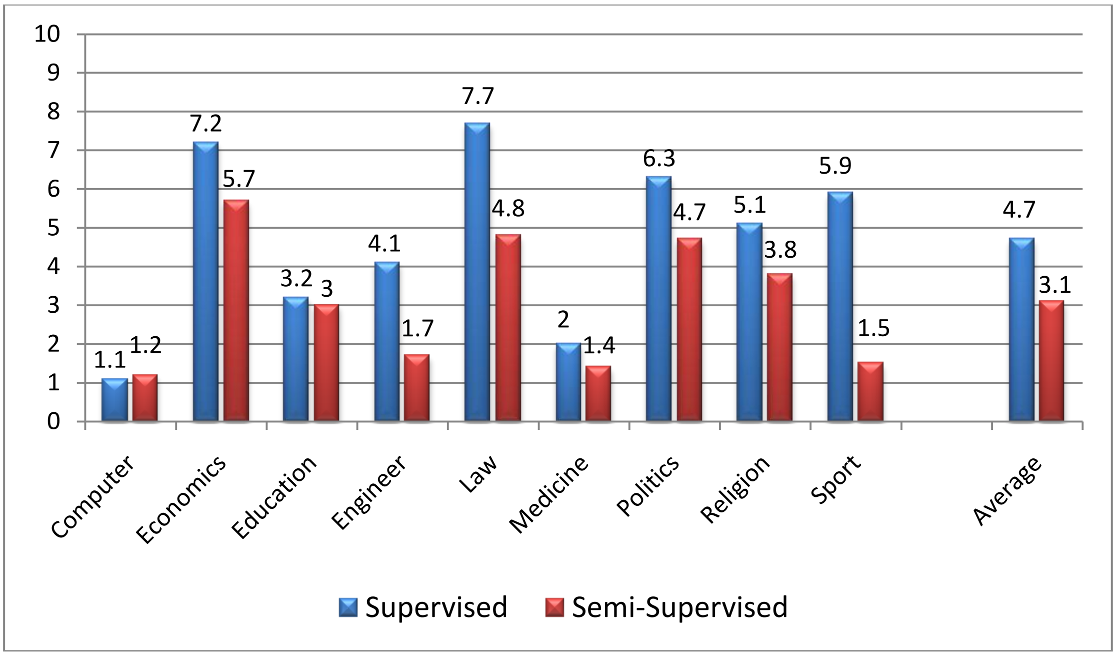


Figure 4.11: Results of experiment 5 for both supervised and semi-supervised methods (error rate).

Table 4.15: Results of experiment 5; accuracy value (recall, precision, fallout, and error rate) for the 9 categories, for both supervised and semi-supervised approaches.

	Supervised				**Semi-Supervised**			
	Re	Pr	Fall	Error	Re	Pr	Fall	Error
Computer	0.88	0.942	0.004	0.011	0.88	0.924	0.005	0.012
Economics	0.79	0.665	0.061	0.072	0.80	0.778	0.035	0.057
Education	0.58	0.836	0.007	0.032	0.62	0.835	0.008	0.030
Engineer	0.83	0.756	0.003	0.041	0.89	0.924	0.008	0.017
Law	0.72	0.508	0.060	0.077	0.75	0.671	0.031	0.048
Medicine	0.81	0.933	0.011	0.020	0.93	0.981	0.003	0.014
Politics	0.76	0.749	0.037	0.063	0.85	0.797	0.031	0.047
Religion	0.87	0.805	0.038	0.051	0.92	0.845	0.030	0.038
Sport	0.63	0.904	0.011	0.059	0.98	0.928	0.012	0.015
Average	0.77	0.773	0.028	0.047	0.87	0.857	0.018	0.031

As shown above, the supervised method gave a very good result, a 77.25% accuracy, which is the same as last experiment. The huge change can be seen in the semi-supervised method as it gave a 86.63% accuracy, which is first, a great result, and second, an amazing improvement over the supervised method. The 20% classified

documents that the algorithm retrained on made almost a 10% difference. This experiment was actually very satisfactory to us and we concluded that retraining the algorithm using the method described above, along with having sufficient number of documents to train on the first time, would result in a highly accurate classifier for Arabic text.

Although this experiment produced very good and satisfactory results, we continued thinking of other ways to further improve our classifier. They are shown in the next two experiments.

Experiment 6

In this experiment, we thought we should try training the algorithm using the 70% labeled data, and then let the algorithm classify the same 70% labeled data, we assumed it should classify most of them correctly but some would be misclassified, and this is what actually happened. The next step was to remove the misclassified documents from the training process, and start the training again using only the ones that were correctly classified. The assumption was that the data that was misclassified may not be very fit for learning purposes, and so removing them could improve the classifiers performance. It turned out as we did that that the performance did not improve and that the accuracy remained the same.

Experiment 7

The other idea that we had was concerned with the way the classifier decides that a document belongs to a certain category. The process of classification is done in the classifying part of the algorithm. The classifier uses naïve Bayes theorem (formula

3.8) to calculate the probability of the document being in each category, and then assigns the category with the highest probability to that document. Our idea was based on the assumption that: in the documents that are classified incorrectly, the highest probability value would be relatively close to the next highest probability value, while in the documents that are classified correctly, the highest probability value would be relatively far from the next highest probability. We assumed that the closeness of the highest two probability values indicated the level of certainty of the choice the classifier makes. For example, if a document has a 45% probability of being "Sport" and a 40% probability of being "Politics", then the classifier would classify it as "Sport" but the level of certainty would not be high. The next choice ("Politics") is pretty close to the first choice, and so we may say that both categories have close-to-equal probabilities. On the other hand, if a document has a 70% probability of being "Sport" and the next highest probability was 10% for "Politics", then the classifier would also classify it as "Sport" but the level of certainty would be high.

We were depending on the assumption that in most misclassified documents, the second highest probability value would be relatively close to the first one, while in most correctly classified documents, the second highest value would be relatively far from the first one. What we intended to do is find the threshold that separates the two groups (group of documents with high certainty, and the one with low certainty) so that we can engage the ones with high certainty in the learning process and exclude the ones with low certainty from it. The ones with low certainty will still be classified as the category with the highest probability, but it will not be added to the training set and therefore will be excluded from the learning process. We do this because we assume that most misclassified documents will have low certainty (and will be

harmful to learn from) and that most correctly classified documents will have high certainty (and will be useful to learn from).

In order to test our idea, we needed to see if the values of the probabilities are as we imagined they would be, so we let the classifier print out all 9 probabilities for each document that it classifies from the unlabeled set. The situation was not as perfect as we thought it would be; many of the misclassified documents had high certainty, and many of the correctly classified documents had low certainty. Still, we did not give up on our idea, so we managed to find three thresholds that may separate the documents into the two clusters. One problem that we faced during this process was that the values of the probabilities were extremely small and very hard to deal with and perform calculations on, so what we did was divide all the values by the value of the highest probability. So now the highest probability = 1, and the rest are all between 0 and 1. According to the observations of the probabilities for all classified documents, we decided to try the values 0.001, 0.01 and 0.1 as thresholds. Let's take 0.1 for example, if the probability of the second choice ≤ 0.1 then it would be considered to be of high certainty and would be included in the learning process, otherwise, it would be considered to be of low certainty and would be excluded.

The results of trying all three thresholds are shown in Figure 4.12.

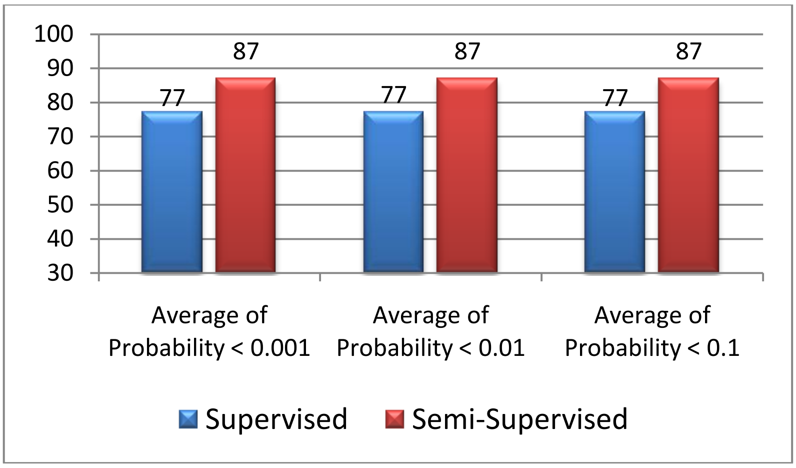


Figure 4.12: Results of experiment 7.

Unfortunately, the results were not as we expected them to be; they did not improve the accuracy of the algorithm, but it was of no big surprise to us because, as mentioned above, the correctly classified and the misclassified documents were not easily distinguished according to the two highest probability values.

Experiment 8

At this point, one issue was still hovering in our minds; will the classifier give a better performance if it was fed more unlabeled documents to classify and then learn from? As we have proven earlier, the algorithm uses unlabeled document to learn and to improve its accuracy, but the fact is, the algorithm will not keep getting better forever; at some point, it will stop improving and will converge to its best performance. So the question is: did the algorithm reach this point, or will its performance improve if it was fed another set of documents?

To answer this question, we had to download yet another set of documents. This new set consisted of 90 documents, 10 per classification. We performed Experiment 1

again, only this time, after the algorithm read, classified, and learned from the 20% unlabeled data, we let the classifier read, classify, and learn from the newly downloaded 90 documents. After all that, the classifier was given the test documents to classify. We compared the results of *Experiment 5* with this experiment (*Experiment 8*) to see if there is any improvement on the classification accuracy of the algorithm; the results showed no improvement and the classification accuracy of the classifier remained the same as the one in *Experiment 5*, which is 87% for the semi-supervised learning approach.

After this experiment, we came to the conclusion that the distribution of the labeled and unlabeled data (70% labeled and 20% unlabeled) in Experiment 5 in addition to the circumstances that the experiment was held in has given the best results for our study, and that these results of the semi-supervised classifier will not show any improvement if the number of unlabeled data given to the classifier was increased. The algorithms performance is at its best under the circumstances of Experiment 5 and has reached its maximum capacity for improvement regarding the number of documents that it can learn from. So, even if the classifier was given more and more documents to learn from, its classification accuracy will not improve any further. And so we concluded that the results of *Experiment 5* shown in Table 4.15 and Figures 4.8, 4.9, 4.10, and 4.11 are the best and most satisfactory results of our study.

4.2.3 Performance

The performances of both supervised and semi-supervised techniques differed in all the previous experiments. We should mention that the classifier spent almost double the time in the semi-supervised learning method than it did in the supervised learning one. The reason for that is that in the supervised learning technique the classifier trains using the labeled data only, while in the semi-supervised learning technique the classifier first trains using the labeled data, and then retrains using both labeled and unlabeled data. So obviously the classifier would be training twice using an even larger set the second time.

Chapter (5): Conclusion and Future Work

In our study, we used the navie Bayes algorithm so serve as our document classifier. Arabic text was targeted due to the importance of this language to us being the language of Qur'an (the holy book of Islam) and our native spoken language. As a result of the richness and uniqueness of this language, complications and difficulties were faced during the text classification process. Techniques, such as preprocessing and stemming, were used to overcome these difficulties, and the classifier was programmed to fit the needs of this language. We used both supervised and semi-supervised learning techniques to train the classifier, and compared between the two in terms of how accurate they can be in classifying new unseen examples. The overall performance of the naïve Bayes algorithm in the text classification problem over Arabic text proved to be very good; the algorithm was very efficient and produced highly accurate results. In the comparison between the supervised and semi-supervised learning techniques, the semi-supervised technique outperformed the supervised one by a good 10% in classification accuracy. We attempted to train the classifier using more documents that we downloaded from online forums, but the classifiers accuracy did not improve any further. And so that was the best performance we could achieve in our study given the dataset we had.

For future work, we suggest the following:

1- Testing other algorithms, such as SVM, *k*-NN, and neural networks in classifying Arabic text using a semi-supervised learning environment.

2- Using multi-words as features in text classification (Zhang, et al., 2008) instead of single-word features as we did in our study.

3- Using naïve Bayes algorithm to predict the diacritic that should be placed over/below the last letter of an Arabic word, having the Holy Qur'an as the training set.

4- Using Arabic text classification to categorize documents into sub-categories in addition to categories.

5- Using virtual examples (Sassano, 2003) to aid in and improve the categorization of Arabic documents into categories and sub-categories.

REFERENCES

Aas K. and Eikvil L. (1999), **Text Categorisation: A survey**, Technical report NR 941, Norwegian Computing Center.

Al-Ameed H., Al-Ketbi S., Al-Kaabi A., Al-Shebli K., Al-Shamsi N., Al-Nuaimi N. and Al-Muhairi S., (2005), **Arabic Light Stemmer: A new Enhanced Approach**, The Second International Conference on Innovations in Information Technology (IIT'05).

Al-Harbi S., Almuhareb A., Al-Thubaity A., Khorsheed M. S., Al-Rajeh A., (2008), **Automatic Arabic Text Classification**, In Proceedings of The 9th International Conference on the Statistical Analysis of Textual Data, Lyon-France.

Al-Kharashi I. And Al-Sughaiyer I. , (2004), **Performance Evaluation of an Arabic Rule-Based Stemmer,** King Abdulaziz City for Science and Technology, Riyadh, Saudi Arabia.

Al-Shalabi R., Kanaan G., and Gharaibeh M. H., (2006), **Arabic Text Categorization Using kNN Algorithm**, Proceedings of The 4th International Multiconference on Computer Science and Information Technology, Vol. 4, Amman, Jordan.

Berger A., Pietra D., and Pietra D., (1996), **A Maximum Entropy Approach to Natural Language Processing**, Computational Linguistics, Vol. 22, No.1, p. 39-71.

Britannica, 2011. Retrieved in April, 2011 from **http://www.britannica.com/EBchecked/topic/31677/Arabic-language**

Ciravegna F., Gilardoni L., Lavelli A., Ferraro M., Mana N., Mazza, S., Matiasek J., Black W. and Rinaldi F. (2000), **Flexible Text Classification for Financial Applications: the FACILE System**, Proceedings of PAIS-2000, Prestigious Applications of Intelligent Systems subconference of ECAI2000, pp. 696-700.

Cogsys, 2010. Retrieved in December, 2010 form **http://cogsys.imm.dtu.dk/thor/projects/multimedia/textmining/node11.html**

Comp, 2011. Retrieved in April, 2011 from **http://www.comp.lancs.ac.uk/computing/research/stemming/general/**

Cortes C. and Vapnik V., (1995), **Support-vector networks**, Machine Learning, 20:273-297.

Dtreg, 2011. Retrieved in April, 2011 from **http://www.dtreg.com/svm.htm**

Duwairi R. M., (2006), **Machine Learning for Arabic Text Categorization**, Journal of the American Society for Information Science and Technology, Vol. 57, No. 8, p. 1005-1010.

Duwairi R. M., (2007), **Arabic Text Categorization**, The international Arab Journal of Information Technology, Vol. 4, No. 2.

El-Halees A. M., (2007), **Arabic Text Classification Using Maximum Entropy**, The Islamic University of Gaza, Journal of Series of Natural Studies and Engineering Vol. 15, No. 1, pp 157-167.

Escalera S., Pujol O., and Radeva P., (2010), **Error-Correcting Ouput Codes Library**, Journal of Machine Learning Research, Vol. 11, p. 661-664.

Gershenson, C., 2003, **Artificial Neural Networks for Beginners**,

Ghahramani, Z., (2004), **Unsupervised Learning**, Gatsby Computational Neuroscience Unit, University College London, UK.

He J. I., Tan A., and Tan C., (2003), **On Machine Learning Methods for Chinese Document Categorization**, Applied Intelligence, vol. 18, pp.311-322.

Joachims T., (1997), **A probabilistic analysis of the rocchio algorithm with t_df for text categorization,** In International Conference on Machine Learning (ICML).

Joachims T., (1998), **Text Categorization with Support Vector Machines: Learning with Many Relevant Features**, Technical Report 23, University of Dortmund, Informatik LS8, Baroper Str. 301, 44221 Dortmund, Germany.

Kivinen J., Warmuth M., and Auer P., (1995), **The perceptron algorithm vs. winnow: Linear vs. logarithmic mistake bounds when few input variables are relevant,** Technical Report In Conference on Computational Learning Theory.

Mitchell T. M., (1997), **Machine Learning**, McGraw-Hill.

Nigam K., McCallum A., Mitchell T. M., 2006, **Semi-supervised Text Classification Using EM**, In Semi-supervised Learning, O. Chapelle, A. Zien, and B. Scholkopf (Eds.), MIT Press.

Nils J. N., (1998), **Introduction to Machine Learning,** An Early Draft Of A Proposed Textbook, Robotics Laboratory, Department of Computer Science, Stanford University.

Peng F., Huang X., Schuurmans D. and Wang S. (2003), **Text Classification in Asian Languages without Word Segmentation**, In Proceedings of the Sixth International Workshop on Information Retrieval with Asian Languages (IRAL 2003), Association for Computational Linguistics, Sapporo, Japan.

Quinlan J. R., (1993), **C4.5: Programs for Machine Learning**, Morgan Kaufmann.

Quora, 2011. Retrieved in April, 2011 from **http://www.quora.com/What-are-popular-text-classification-algorithms-in-commercial-use-and-how-are-they-used**

Rennie J. D. M. and Rifkin R., (2001), **Improving Multiclass Text Classification with the Support Vector Machine,** Technical Report, Artificial Intelligence Laboratory, MIT, Cambridge, Massachusetts

Resample, 2011. Retrieved in April, 2011 from **http://www.resample.com/xlminer/help/NaiveBC/classiNB_intro.htm**

Rish, I., 2001, **An Empirical Study of the Naive Bayes Classifier**, IJCAI, Workshop on Empirical Methods in Artificial Intelligence.

Rocchio J., (1971), **Relevance Feedback in Information Retrieval**, In G. Salton, editor, The SMART Retrieval System: Experiments in Automatic Document Processing, pages 313-323, Prentice-Hall Inc..

Saravananthirumuruganathan, 2011. Retrieved in April, 2011 from **http://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/**

Sassano M., (2003), **Virtual Examples for Text Classification with Support Vector Machines**, Fujitsu Laboratories Ltd, Kawasaki, Japan.

Scholarpedia, 2010. Retrieved in December, 2010 from **http://www.scholarpedia.org/article/Text_categorization**

Stackoverflow, 2011. Retrieved in April, 2011 from **http://stackoverflow.com/questions/1832076/what-is-the-difference-between-supervised-learning-and-unsupervised-learning**

Statsoft, 2010. Retrieved in December, 2010 from **http://www.statsoft.com/textbook/naive-bayes-classifier**

Tsay J. J. and Wang J. D., (2004), **Improving Linear Classifier for Chinese Text Categorization**, Information Processing and Management, vol. 40, no. 2, pp.223-237.

Vapnik V. N, (1995), **The Nature of Statistical Learning Theory**, Springer, New York.

Videolectures, 2010. Retrieved in December, 2010 from **http://videolectures.net/mlas06_mitchell_sla/**

Welling, M., **Support Vector Machines**, Department of Computer Science, University of Toronto, Toronto, M5S 3G5 Canada.

Yang Y., (1997), **An evaluation of statistical approaches to text categorization**, Technical Report CMU-CS-97-127, Carnegie Mellon University.

Zhang W., Yoshida T., Tang X., (2008), **A Study With Multi-Word Features in Text Classification**, School of Knowledge Science, Japan Advanced Institute of Science and Technology, Japan.

Zhou X., Zhang X., and Hu X., (2008), **Semantic Smoothing for Bayesian Text Classification with Small Training Data**, to appear in 2008 SIAM International Conference on Data Mining (SDM2008), April 24-26, Atlanta, Georgia.

Zhu, X., 2008, **Semi-supervised Learning Literature Survey**, Computer Sciences TR 1530, University of Wisconsin – Madison.

تقنيات شبه-موجهة لتصنيف النصوص العربية

إعداد
أمان موفق عبدالله حجة

المشرف
الدكتور خليل الهندي

المشرف المشارك
الدكتور ثائر حمتيني

ملخص

إن الكم الهائل من المعلومات النصية و الزيادة المستمرة في عدد النصوص الالكترونية هو أحد أسباب اللجوء الى استخدام التصنيف الالكتروني للنصوص. لقد قمنا باختيار خوارزمية "بايز البسيطة" في هذه الدراسة (واحدة من خوارزميات التعلم الآلي) لتقوم بالتصنيف الالكتروني للنصوص. هذه الخوارزمية كانت قد أثبتت جدارة في مجال تصنيف النصوص بالإضافة إلى أنه قد وجد أنها تعطي نتائج جيدة جدا و بدقة تصنيف عالية. كان اختيارنا في هذه الدراسة قد وقع على اللغة العربية و ذلك لأننا لم نجد أي دراسات سابقة استخدمت نصوص عربية في تقنية التعلم الشبه-موجه. هذه اللغة، كما هو الحال في أي لغة أخرى، هي لغة غنية و مميزة بطريقتها الخاصة و لديها خصائصها الخاصة بها. لقد تم التطرق لقضايا و خصائص اللغة العربية في هذه الدراسة و قمنا بتعديل المصنف ليتوافق مع احتياجات هذه اللغة. لقد قمنا بتعليم المصنف باستخدام تقنية التعلم الموجه و تقنية التعلم الشبه-موجه. أحد أهم أهداف هذه الدراسة كان لاكتشاف اذا ما كانت مهمة تصنيف النصوص العربية ستتحسن نتيجة استخدام تقنية التعلم الشبه-موجه، لذلك قمنا باختبار المصنف بعد استخدام كل من التقنيتين و وثقنا النتائج. مجموعة البيانات الخاصة بنا تتكون من 1893 نص كتبوا باللغة العربية، هذه النصوص تنتمي الى 9 مجموعات: علم الحاسوب، اقتصاد، تعليمي، هندسة، قانون، طب، سياسة، ديني، و رياضة. لقد قمنا بتقسيم مجموعة البيانات الى ثلاث مجموعات: البيانات المصنفة، البيانات الغير مصنفة، و البيانات المخصصة للاختبار. 70% من البيانات شكلت من البيانات المصنفة، 20% من البيانات شكلت من البيانات غير المصنفة، و 10% من البيانات شكلت من البيانات المخصصة للاختبار. لقد قمنا باستخدام البيانات المصنفة والبيانات المخصصة للاختبار في تقنية التعلم المجه، بينما استخدمنا البيانات المصنفة و البيانات المخصصة للاختبار بالاضافة الى البيانات الغير مصنفة في تقنية

التعلم الشبه-موجه. أظهرت نتائج تقنية التعلم الموجه دقة تصنيف وصلت الى 77% بينما كانت نتائج تقنية التعلم الشبه-موجه 87%. أثبتت تقنية التعلم الشبه-موجه انها تحقق نتائج ذات دقة تصنيف أعلى من تلك التي تحققها تقنية التعلم الموجه عند استخدمها لتصنيف النصوص العربية بفرق يصل الى 10%. لقد أثبتت الدراسة أيضا بأن نتائج استخدام تقنية التعلم الشبه-موجه لن تتحسن اذا قمنا بزيادة عدد النصوص الغير مصنفة و انها قد وصلت الى الحد الذي تعطي فيه أفضل النتائج.